STIC Search Report
EIC 2100

STIC Database Tracking Number: 169068

TO:	From:
Michael Pham **Location: RND-3D18** **Art Unit: 2167** **Thursday, January 26, 2006** **Case Serial Number: 10/614,527**	**Lance Sealey** **Location: EIC 2100** **RND-4B11** **Phone: 571-272-8666** **Lance.Sealey@uspto.gov**

Search Notes

Dear Michael,

Here are the results of your search request.
Please let me know if you have any questions.

Lance

STIC EIC 2100 Search Request Form

176769

65

Today's Date: 1/17/2006

What date would you like to use to limit the search?

Priority Date: 7/7/2003 Other:

Name Michael Pham

AU 2165 Examiner # 81563

Room # 3D18 Phone 23924

Serial # 10/614527

Format for Search Results (Circle One):

PAPER DISK EMAIL

Where have you searched so far?

USP DWPI EPO JPO ACM IBM TDB

IEEE INSPEC SPI Other EAST

Is this a "Fast & Focused" Search Request? (Circle One) YES (NO)

A "Fast & Focused" Search is completed in 2-3 hours (maximum). The search must be on a very specific topic and meet certain criteria. The criteria are posted in EIC2100 and on the EIC2100 NPL Web Page at http://ptoweb/patents/stic/stic-tc2100.htm.

What is the topic, novelty, motivation, utility, or other specific details defining the desired focus of this search? Please include the concepts, synonyms, keywords, acronyms, definitions, strategies, and anything else that helps to describe the topic. Please attach a copy of the abstract, background, brief summary, pertinent claims and any citations of relevant art you have found.

Database Integrity

Concept: Attached

Essentially, seeking a database integrity system utilizing hash functions in order to verify the integrity of the db.

Claim 1 is most important.

SEARCH PREP TIME: 100 MIN.
TERMINAL TIME: 380 MIN.

RECEIVED
JAN 17 2006
BY:

STIC Searcher LANCE SEALEY Phone 2-8666

Date picked up 1/24/06 Date Completed 1/25/06

sira Search and Information Resources Administration

US005819291A

United States Patent [19]

Haimowitz et al.

[11] Patent Number: 5,819,291

[45] Date of Patent: Oct. 6, 1998

[54] **MATCHING NEW CUSTOMER RECORDS TO EXISTING CUSTOMER RECORDS IN A LARGE BUSINESS DATABASE USING HASH KEY**

[75] Inventors: **Ira Joseph Haimowitz**, Niskayuna; **Brian Terence Murren**, Clifton Park; **Henry Lander**, Niskayuna; **Barbara Ann Pierce**, Slingerlands; **Mary Clarkeson Phillips**, Delmar, all of N.Y.

[73] Assignee: **General Electric Company**, Schenectady, N.Y.

[21] Appl. No.: **702,379**

[22] Filed: **Aug. 23, 1996**

[51] **Int. Cl.**[6] **G06F 17/30**

[52] **U.S. Cl.** **707/201**; 707/1; 707/5

[58] **Field of Search** 707/201, 5, 3, 707/10, 2

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,290,105	9/1981	Cichelli et al.	707/5
5,129,082	7/1992	Tirfing et al.	707/3
5,202,982	4/1993	Gramlich et al.	707/2
5,446,888	8/1995	Pyne	707/10
5,479,654	12/1995	Squibb	707/201
5,481,704	1/1996	Pellicano	707/5
5,604,910	2/1997	Kojima et al.	707/3

OTHER PUBLICATIONS

SSA–Name An Introduction', Search Software America, Old Greenwich, CT, Version 1.6, pp. 1–24, Oct. 1, 1994.

"The Field Matching Problem: Algorithms And Applications" by AE Monge, Et Al, Second International Conference On Knowledge Discovery And Data Mining, Aug. 2, 1996, AAA1 Press, Menlo Park, Ca, pp. 267–270.

"The Merge/Purge Problem for Large Databases" by MA Hernandez, et al, Proceedings of the ACM SIGMOID International Conference on Management of Data (May 1995), pp. 1–35.

Primary Examiner—Thomas G. Black

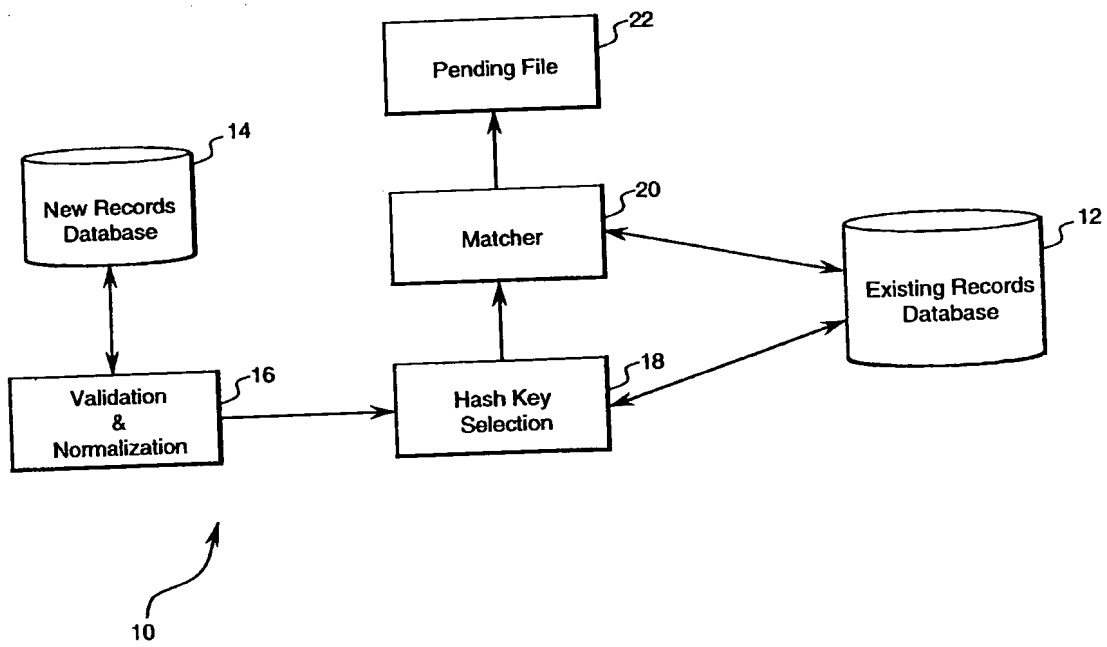
Assistant Examiner—Srirama Channavajjala

Attorney, Agent, or Firm—David C. Goldman; Marvin Snyder

[57] **ABSTRACT**

In this invention there is a method and system for matching new customer records to existing customer records in a database. The new customer records are validated for quality and normalized into a standard form. A hash key is selected to generate a candidate set of records from the existing records in the database that likely matches the new customer records. The new customer records are then matched to each of the records in the candidate set. Once the matching has been performed, a decision is made on whether to create a new customer record, update an existing record, or save the new record in a pending file for resolution at a later time. In another embodiment, there is a methodology for learning matching rules for matching records in a database. The matching rules are then used for matching a new customer record to existing records in a database.

18 Claims, 6 Drawing Sheets



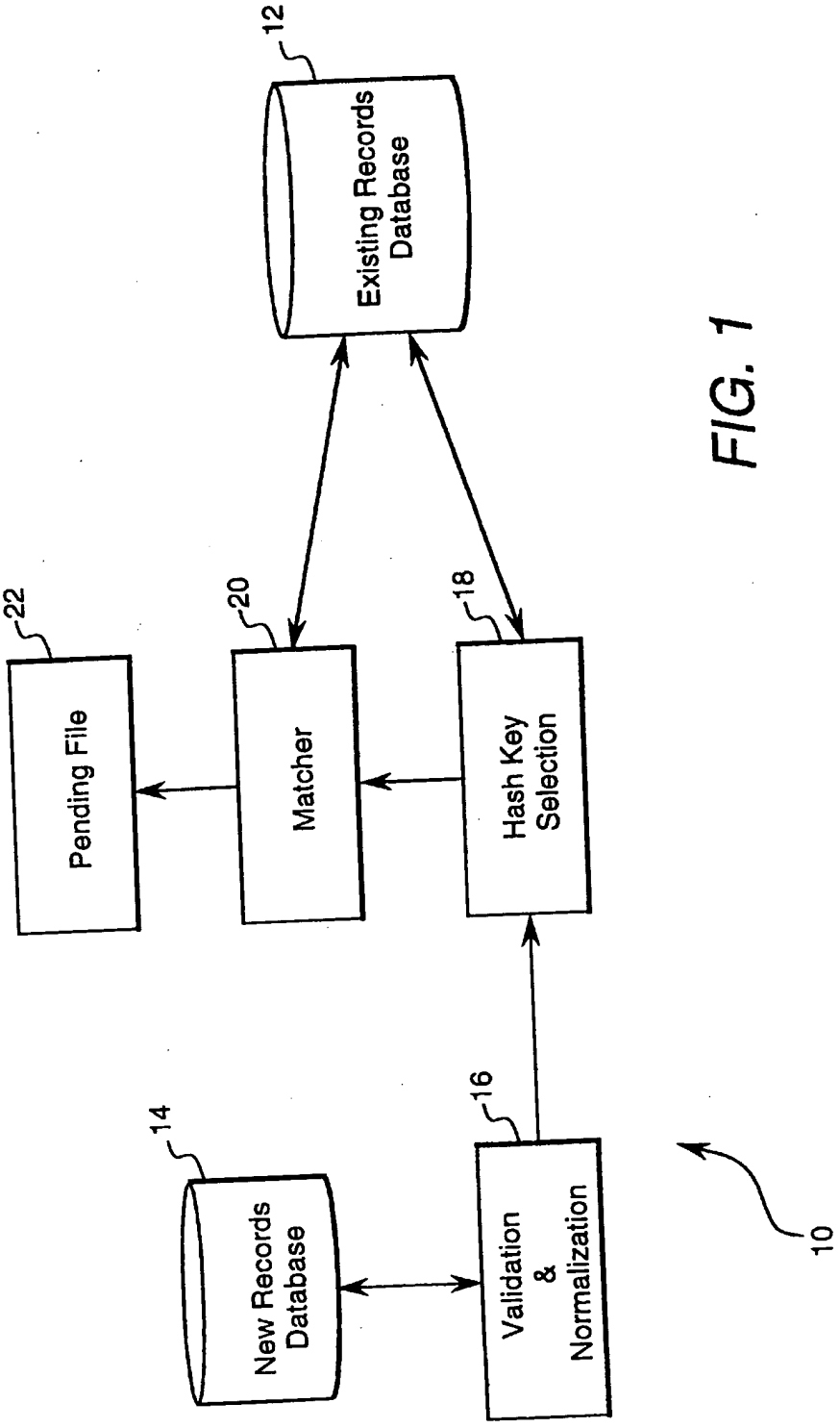
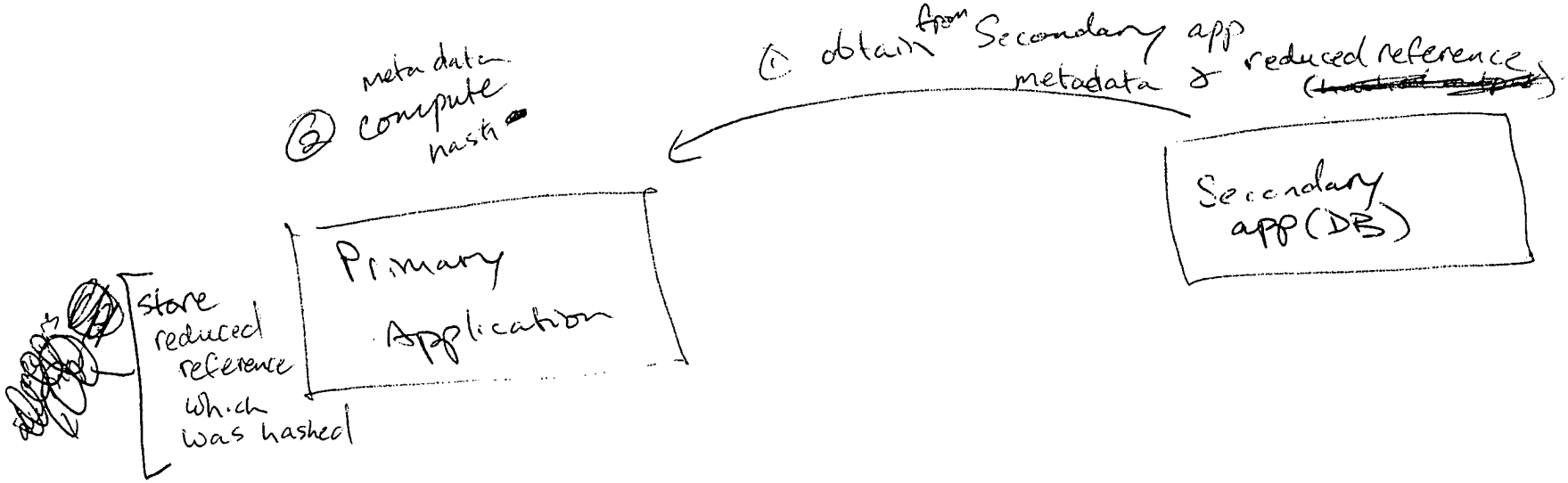

12
Existing Records Database
22
Pending File
20
Matcher
18
Hash Key Selection
14
New Records Database
16
Validation & Normalization
10

FIG. 1



④ Compare Stored meta hash vs. stored hash ref.

⑤ if matches, access DB
If not stop and send message to user.

Essentially figure 8 of app. is a good example.

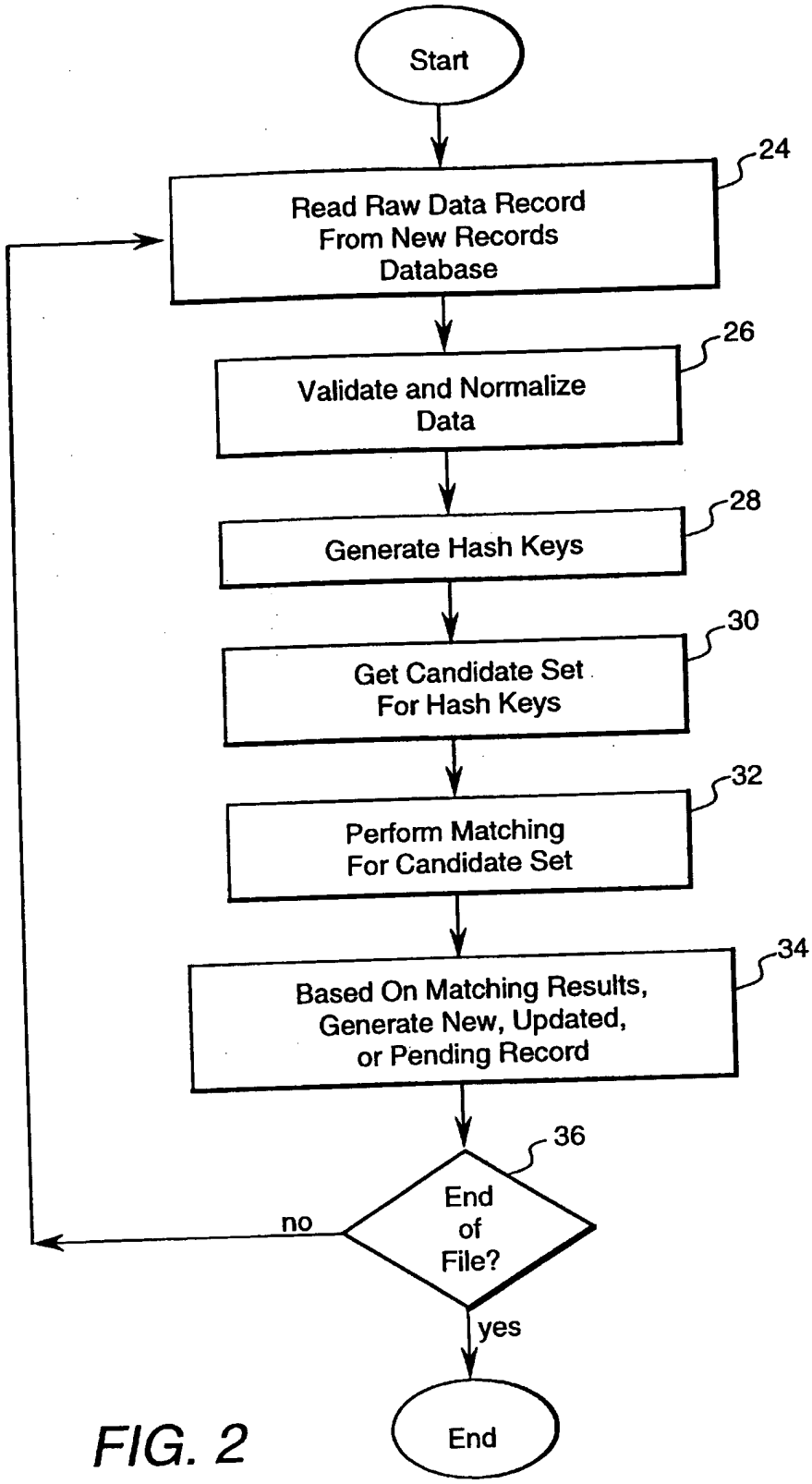

Start
Read Raw Data Record From New Records Database
24
Validate and Normalize Data
26
Generate Hash Keys
28
Get Candidate Set For Hash Keys
30
Perform Matching For Candidate Set
32
Based On Matching Results, Generate New, Updated, or Pending Record
34
End of File?
36
no
yes
End

FIG. 2

Field No.	Field Name	Width	Description
1	Customer Name	50	Customer Name
2	Address Line 1	25	1st Address Line
3	Address Line 2	25	2nd Address Line
4	Address Line 3	25	3rd Address Line
5	Address Line 4	25	4th Address Line
6	Cotu	25	City Name
7	State or Province	2	US State or Canadian Province
8	Postal	9	Postal/Zip Code
9	Country Number	3	Standard ISO Country Code
10	Customer Phone Number	15	Phone Number of Customer
11	Duns Location Number	9	Unique Duns Number of Customer
12	Ultimate Duns Number	9	Duns Number of Parent Company
13	Ultimate Name	50	Name of Parent Company
14	VAT Number	15	Value-Added Tax Number of Custome
15	Federal Taxpayer ID	9	US Federal Taxpayer ID Number of Customer
16	Primary SiC Code	4	Standard Industry Code of Customer

38

FIG. 3

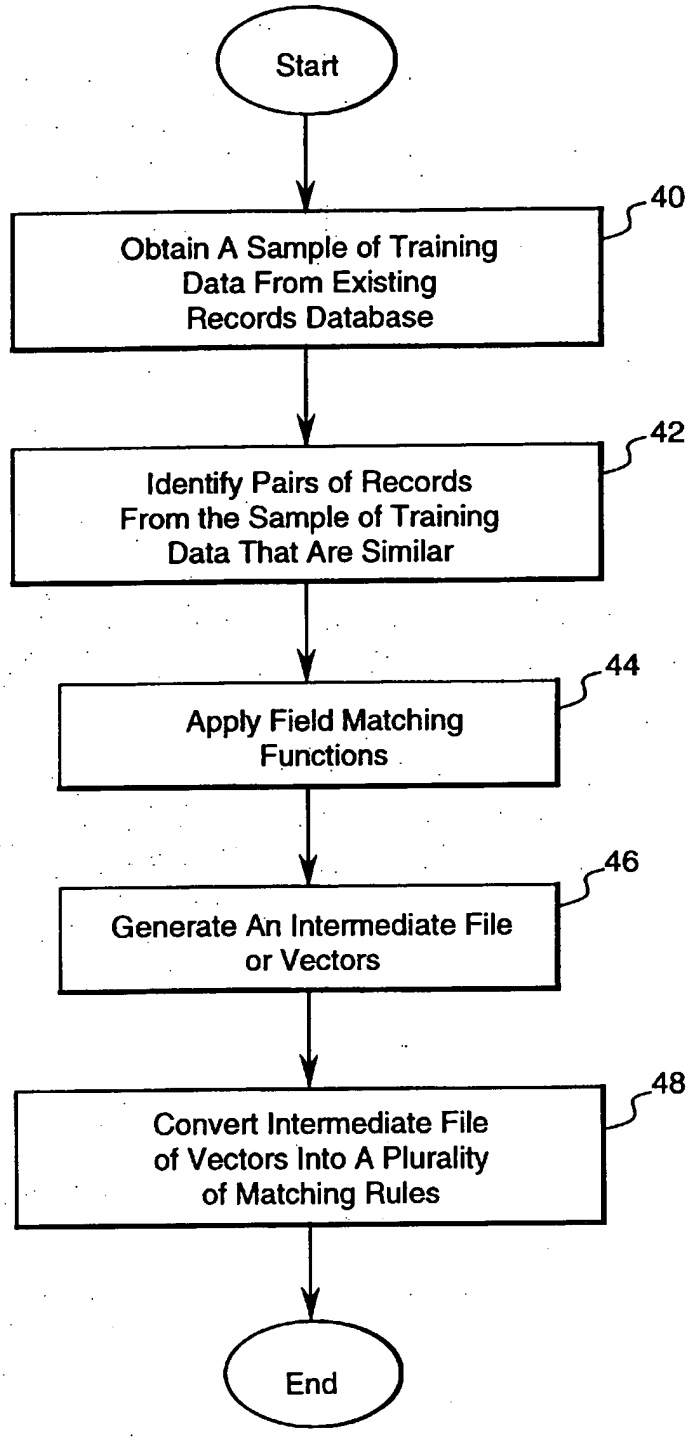

Start
Obtain A Sample of Training Data From Existing Records Database
40
Identify Pairs of Records From the Sample of Training Data That Are Similar
42
Apply Field Matching Functions
44
Generate An Intermediate File or Vectors
46
Convert Intermediate File of Vectors Into A Plurality of Matching Rules
48
End

FIG. 4

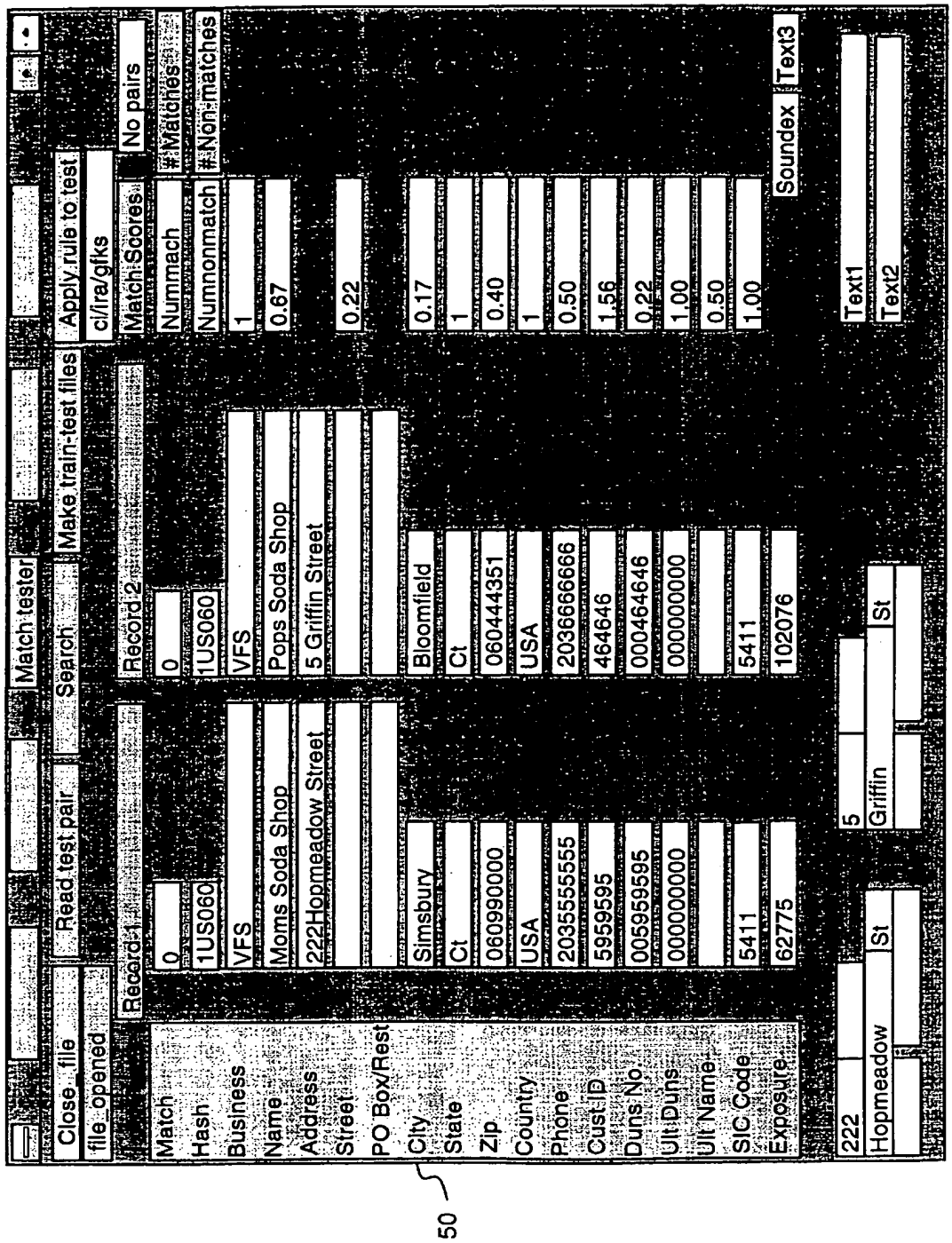

Match tester
Close file
Read test pair
Search
Make train-test files
Apply rule to test
file opened
Record 1
Record 2
Match Scores
No pairs
Match
Hash
Business
Name
Address
Street
PO Box/Rest
City
State
Zip
Country
Phone
Cust ID
Duns No
Ult Duns
Ult Name
SIC Code
Exposure
Nummach
Numnonmatch
Matches
Non matches
1US060
VFS
Moms Soda Shop
222Hopmeadow Street
Simsbury
Ct
060990000
USA
2035555555
5959595
005959595
000000000
5411
62775
1US060
VFS
Pops Soda Shop
5 Griffin Street
Bloomfield
Ct
060444351
USA
2036666666
464646
000464646
000000000
5411
102076
0.67
0.22
0.17
1
0.40
1
0.50
1.56
0.22
1.00
0.50
1.00
Soundex
Text3
Text1
Text2
222
Hopmeadow
St
5
Griffin
St
50

FIG. 5

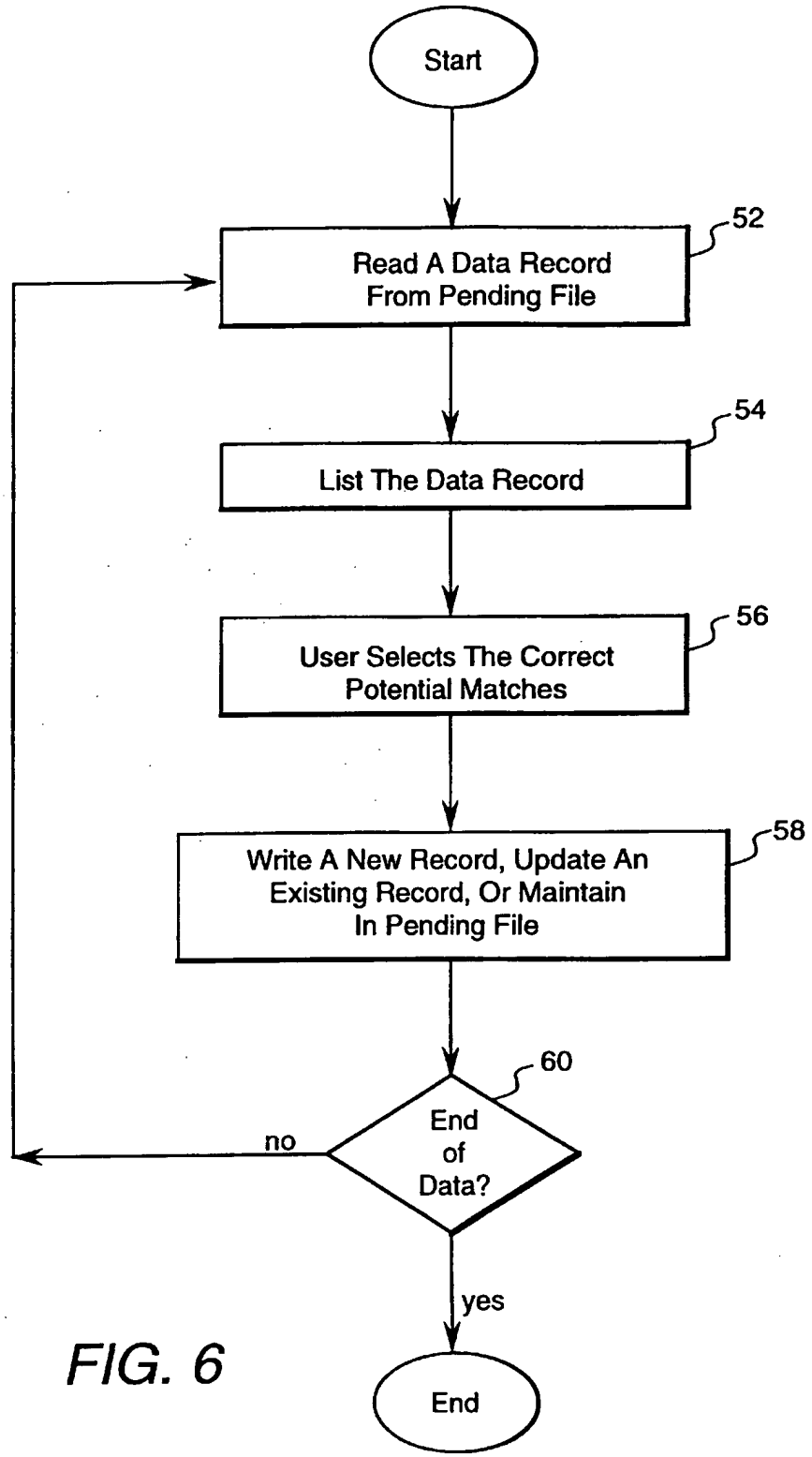

Start
Read A Data Record From Pending File
52
List The Data Record
54
User Selects The Correct Potential Matches
56
Write A New Record, Update An Existing Record, Or Maintain In Pending File
58
End of Data?
60
no
yes
End

FIG. 6

MATCHING NEW CUSTOMER RECORDS TO EXISTING CUSTOMER RECORDS IN A LARGE BUSINESS DATABASE USING HASH KEY

BACKGROUND OF THE INVENTION

The present invention relates generally to databases, and more particularly to matching new customer records to existing customer records in a large business database.

A large business database often has duplications of the same customer records. The duplications are likely due to misspelling errors or because of multiple methods of entering the customer records into the database. These duplications result in several problems for the end-user. One problem is that a customer whose records have been duplicated may receive multiple mailings from the end-user. Another problem is that the end-user may not ever have consistent information about each customer. The customer information may be inconsistent because every time the customer record has to be updated, only one record is updated. There is no assurance that the most recently updated record will be revised, which results in inconsistent information. A third problem with duplicated records, is that the end-user is unable to determine how much business activity has been generated by a particular customer.

Current methods of searching a large business database is performed by library-style catalogue retrieval systems. These library-style catalogue retrieval systems can search a large database of records to find matches that are similar to a query entered by an end-user. Typically, these library-style catalogue retrieval systems use phonetic-based algorithms to determine the closeness of names or addresses or word strings. A problem with these library-style catalogue retrieval systems is that they are only useful for searching through an existing customer database and are unable to compress a large customer database having multiple repetitions of customer records. Therefore, there is a need for a methodology that processes new customer records, checks the new records for poor quality, normalizes and validates the new records, and matches the new records to existing customer records in order to determine uniqueness. Normalizing, validating, and matching the customer records will allow an end-user to avoid wasted mailings, maintain consistent information about each customer, and determine how much business activity has been generated by a particular customer.

SUMMARY OF THE INVENTION

Therefore, it is a primary objective of the present invention to provide a method and system that normalizes and validates new customer records, and matches the new records to existing customer records in a large database.

Another object of the present invention is to enable end-users of large business databases to avoid wasted mailings, maintain consistent information about each of their customers, and determine how much business activity has been generated by a particular customer.

Thus, in accordance with the present invention, there is provided a method and a system for matching a new data set containing a record and a collection of fields to an existing data set in a database containing a plurality of records each having a collection of fields. In this embodiment, the new data set is initially read. Each of the fields from the record in the new data set are then validated. The validated fields in the record in the new data set are then normalized into a standard form. Next, a hash key is selected for generating a candidate set of records from the existing data set in the database that likely matches the record from the new data set. The hash key is then applied to the plurality of records in the existing data set of the database to generate the candidate set of records. The record from the new data set is then matched to each of the records in the candidate set. The existing data set in the database is then updated according to the results of the match between the record from the new data set to the records in the candidate set.

In accordance with another embodiment of the present invention, there is provided a method and system for generating rules for matching data in a database containing a plurality of records each having a collection of fields. In this embodiment, a sample of training data is obtained from the database. Similar pairs of records from the sample of training data are then identified. Field matching functions are applied to each of the corresponding fields in the similar pairs of records. Each field matching function generates a score indicating the strength of the match between items in the field. An intermediate file of vectors containing matching scores for all of the fields from each of the similar pair of records is then generated. The intermediate file of vectors are then converted into a plurality of matching rules for matching data in the database. The plurality of matching rules can then be used for matching a new data set containing a record and a collection of fields to an existing data set in a database containing a plurality of records each having a collection fields.

While the present invention will hereinafter be described in connection with a preferred embodiment and method of use, it will be understood that it is not intended to limit the invention to this embodiment. Instead, it is intended to cover all alternatives, modifications and equivalents as may be included within the spirit and scope of the present invention as defined by the appended claims.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a system for data validation and matching according to the present invention;

FIG. **2** is a flow chart describing the data validation and matching according to the present invention;

FIG. **3** is an example of a fixed general business file format that may be used in the present invention;

FIG. **4** is flow chart describing the matching process in more detail;

FIG. **5** is a screen view of an interface used for the matching process; and

FIG. **6** discloses a flow chart describing the process of examining pending data for a match.

DETAILED DESCRIPTION OF THE PRESENT INVENTION

The present invention discloses a data validation and matching tool for processing raw business data from large business databases. The raw business data includes a plurality of records each having a collection of fields and attributes. A block diagram of a system **10** for performing data validation and matching according to the present invention is shown in FIG. **1**. The system **10** includes a database of existing customer records **12** and a database of new customer records **14**. The database of existing customer records **12** can be a large database containing over 200,000 records. Each record has a collection of fields and attributes that are applicable for the particular business application. For example, some of the fields that may be used are

business name, customer name, address, country, phone number, business codes, etc. The database of new customer records **14** can be as small as one record or as large as over 200,000 records. These records also have a collection of fields and attributes that are applicable to the business application.

The system **10** also includes a computer such as a workstation or a personal computer. A representation of the functions performed by the computer are shown as blocks **16, 18, 20,** and **22**. In particular, a validation and normalization block **16** reads the data from the new records database **14** and checks the fields in each record for quality and normalizes the field information into a standard form. If the data is good, then a hash key selector **18** selects a hash key. Note that there may be one or more hash keys. A matcher **20** uses the hash key to select a set of candidates from all of the existing records in the database **12** with the same hash key. For example, the present invention will generate about 100 candidates for a 50,000 record database. The matcher **20** performs a matching operation between a new data record from database **14** and each member of the candidate set. The matching operation, which is described below in further detail, creates a list of potential matches. If multiple hash keys are used, then the process will retrieve records based on a disjunction of the hash keys. However, once all the matching is done, the matcher **20** makes a decision whether to create a new customer record in database **12**, update an existing record in database **12**, or save the new data in a pending file **22** for resolution at a later time.

FIG. **2** is flow chart describing the operation of the data validation and matching according to the present invention. The operation begins by reading raw data from a record at **24**. The data from the record is validated for quality and standardized into a standard form at **26**. Hash keys are selected at **28** by the hash key selector **18**. At **30**, a set of candidates from all of the existing records in the database **12** with the same hash key are retrieved. The matching operation is then performed at **32** between the new data record and each member of the candidate set, resulting in a list of potential matches. Based on the matching results, block **34** creates either a new record in database **12**, or updates an existing record in database **12**, or places the new record in a pending file for resolution at a later time. If there are more records in the raw data file at **36**, then the next record is read and the steps at **26, 28, 30, 32,** and **34** are repeated. The operation ends once there are no more records to be processed.

Before validation and normalization, the raw data file from the new records is read. In the present invention, the data can arrive from many different hardware/software systems ranging from an off-the-shelf spreadsheet application to a mainframe dump that is in a fixed "General Business File" format. An example of a fixed general business file format **38** is shown in FIG. **3**. The business file format includes a field number, field name, width of the field, and description of the field. Note that the business file format in FIG. **3** is only an example of possible fields and can change depending upon the particular business application. For instance, files of hospital patients may include patient name, date of birth, hospital ID, and patient sex.

After the raw data file has been read, each of the data fields in a record are examined to see if they can be validated. Data fields that contain internal codes are validated against a validation code table. The present invention can then download a SIC (Standard Industrial Classifications) table and a Zip Code table obtained from the United States Postal Office and check for validation. Other validation tables containing country codes and various abbreviations can be used as other validation tables. To speed up processing, all of the validation tables are loaded into memory at the beginning of the running of the data validation and matching system **10**, with the exception of any zip code table which is generally too large. Then each of the fields from the record in the new data set are compared to the standards set forth in the validation tables and verified to make sure that the data corresponds to the standards listed in the validation table. In the present invention, the country code field is tested before the zip code field. From the zip code table it is possible to verify city and state. If there is only a city and state with no zip code, then the zip can be dubbed. The DUNS (a unique number assigned by Dun & Bradstreet Corp.) number can be validated using Dun & Bradstreet Corp. Modulus Ten Check Digit™ algorithm for the ninth digit. The data validation and matching system **10** also can check for clearly erroneous phone numbers (e.g., all 53s, or all 93s).

Once the data has been validated, the next step is to normalize the name and address. The idea behind normalizing the incoming data is to standardize names and addresses to facilitate matching without diluting the name or address. Standardization also enables better end-user name searches in the future. In the present invention, the normalization step removes excess punctuation such as periods and commas, leaving other punctuation from the name and address fields to avoid dilution. Abbreviations are handled in a different manner. For example, in some cases, an end-user types CORPORATION while someone else types CORP, or ST versus STREET. To avoid over-abbreviation and dilution the present invention uses a combination of abbreviations and expansions. In particular, the raw data is examined for the most common cases of alternative spellings. In addition, other items that are not part of the street address such as ATTN: ACCOUNTS PAYABLE are removed.

If the data in the fields are bad or unsuitable for normalization, then the record is removed from further consideration and placed in the pending file **22**. The rules for determining bad or unsuitable data are very straightforward. In particular, if part of the country code and part of the postal code (zip code relates to US addresses and postal code relates to international addresses) cannot be used as a hash key, then the data is bad and is placed into a "bad data" file for future resolution. The rules for insertion into the bad data file are as follows:

1) Lack of address - blank country - state - city - zip;

2) No way to determine country; and

3) In the US: no zip code and cannot derive the zip code from city and state.

Rule 1 states that the data will be placed in the bad data file if it lacks an address such as country, state, city, and zip code. Rule 2 states that the data will be placed in the bad data file if there is no way to determine the country. Rule 3 states that the data will be placed in the bad data file if the country of the data is the U.S. and there is no zip code or the zip code cannot be derived from the city and state. Other bad data rules may be used for other business applications.

Once the customer record under consideration has been normalized, the next step is to determine if it matches any of the existing customers in the existing records database **12**. However, given the size of the existing records database **12**, it is too time consuming to perform a match of every record against the new customer records. Thus, a hash key is used to identify the set of existing customer records (i.e., the candidate set) against which the new record under consid-

eration will be matched. The hash key is a field that is added to the customer files expressly for the purpose of identifying the candidate set. This field contains a concatenation of data in the other general business format fields. The specific composition of the hash key field is intended to maximize the likelihood that if the new customer record under consideration is in the database **12**, then it will be found in the generated candidate set.

Although the illustrative embodiment of the present invention uses a single hash key, it is within the scope of the invention to utilize more than one hash key as part of the candidate set generation. Multiple hash keys increase the likelihood of correct classification by providing alternative paths for locating the matching record. These alternative paths can be used to locate the customer record in the database **12** if the record fields from which the primary hash key is computed contain invalid or missing data. However, the use of multiple hash keys increases implementation complexity and adversely impacts execution speed.

In the illustrative embodiment of the present invention, it is given that customer classification is based on a corporate entity located at a particular physical location. And it follows that the hash key used to identify the possible matches should generate the set of similar corporate entities located in a similar location. Thus, the hash key should be composed of attributes that describe the corporate entity and ones that describe its physical location. In general, when determining the combination of attributes to be used for the hash key it is important that the hash key be able to generate a list of likely matches (i.e., similar companies located in a similar location to the one being classified). Also, the hash key should be made up of fields containing the highest quality of data available. Data quality refers to the degree to which data is present and valid. Furthermore, the hash key should not generate candidate sets that are excessively large. This will impact subsequent matching speed. Another consideration is that the hash key should generate appropriate candidate sets for both domestic as well as international companies. Also, the hash key should not be overly lengthy, because it will impact the search speed of the database. Finally, the hash key should not be overly restrictive, because if the matching customer record is not included in the candidate set, then the match will not be made. Using the general business format provided in FIG. **3**, the NAME field is the only Corporate Entity attribute that contains data of sufficient quality to warrant consideration for hash key inclusion. Of the Location attributes, the CITY, STATE, ZIP, and COUNTRY fields contain the highest quality data for consideration of a hash key.

Using the above considerations, one possible hash key function is:

SUBSTR(NAME,1,1)+COUNTRY_CODE+SUBSTR(ZIP,1,3).

This hash key function is composed of the first letter of the NAME field, a two-character COUNTRY_CODE, and the first three characters of the ZIP CODE field. This hash key function works well because it is composed of high quality attributes, it produces relatively small candidate sets, and it is very compact. Another possible hash key function is:

SUBSTR(NAME,1,1)+COUNTRY_CODE+SUBSTR(CITY,1,7).

This hash key function is composed of the first letter of the NAME field, a two-character COUNTRY_CODE, and the first seven characters of the CITY field. This hash key function works well because it is composed of high quality attributes, it is composed of attributes that are particularly useful when dealing with international customer records, and it is very compact. When the two hash key functions are used in combination with each other, a broadened search strategy results. For example, one possible hash key that could be used is IUS281. This hash key has returned a candidate set of four records from the existing records database **12**, which has over 200,000 records. This candidate set dramatically reduces the scope of the matching task.

After the candidate set has been retrieved, the matcher **20** determines if there is a match between the data of the new record set and records of the candidate set. In particular, the matcher **20** compares the data from the new record set to each of the records in the candidate set. Each comparison generates a matching score indicating a degree of match between the record from the new data set and each record in the candidate set. Depending upon the results of the matching, the matcher **20** makes a decision whether to create a new customer record in database **12**, update an existing record in database **12**, or to save the new data in the pending file **22** for resolution at a later time.

In order for the matcher **20** to determine if there is a match, it is necessary that the matcher learn a plurality of matching rules beforehand. FIG. **4** discloses a flow chart describing how the plurality of matching rules are learned in the present invention. The plurality of matching rules are learned by first obtaining a sample of training data from the existing records database **12** at **40**. Next, pairs of records from the sample of training data that are similar are identified at **42**. Field matching functions are then applied to each of the corresponding fields in the similar pairs of records at **44**. Each field matching function generates a score indicating the strength of a match between items in the corresponding fields. Next, an intermediate file of vectors is generated at **46**. The intermediate file of vectors contains matching scores for all of the fields from each of the similar pair of records. The intermediate file of vectors is then converted into a plurality of matching rules for matching the data in the databases at **48**.

As mentioned above, the process of learning matching rules begins by examining similar pairs of records from the existing records database **12** and labeling the pairs of records as a "match" or "non match." This step is needed to learn automatically what attributes contribute to a match and what do not. The next step is to apply field matching functions to each of the corresponding fields in the similar pairs of records. In the illustrative embodiment, there are about a dozen main attributes of a customer record that are matched. Each attribute uses a field matching function to generate a score indicating the strength of the match. In particular, each field matching function takes two strings and computes a matching score from 0 for a non match to a 1 for a perfect match. In the present invention, there are three types of field matching functions. One type of field matching function is the exact match of strings between attributes. The exact field matching function is preferably used for the country and business attributes. Another type of field matching function is the phonetic-based match of attributes such as NAME and ADDRESS. The phonetic-based match may performed by using an off-the-shelf software application such as Soundex™, which matches phonetically similar names (e.g. Johnson, Johansen, Jonson, etc.) by compressing each input string, removing vowels, and comparing the compressed strings. Attributes such as the zip code, phone number, customer ID, Duns, Ultduns, and SIC are matched based on a character by character string comparison. In most cases, a

field matching function is built in for each attribute. For example, for zip code matching, 5 character codes and 9 character codes were handled differently, and an extra weight was given to the 4th digit.

Sometimes extraneous elements or omissions in the address string offset the string, yielding inaccurate match scores. In order to obtain more accurate matches the present invention has adopted a strategy that exploits the inherent structure in an address by parsing the address into separate components and fitting them into certain templates. The basic templates are:

PO BOX - Number - Street name - Street descriptor - Direction;

PO BOX - Number - Direction - Street name - Street descriptor;

PO BOX - Number - Direction - Street name;

where any of the components in the templates could be missing. Note that the normalizing procedure standardizes the street descriptors, directions, and numbers prior to address matching. Following the parsing, the address field matching function generally proceeds as follows. The "PO BOX" is extracted first and then the input string is traversed to the end until the street descriptor is located. Next, the street number is searched until it is found. Then the direction element is checked and the street descriptor. All elements between the descriptor and the direction, or number are considered to be the street name.

Following the template fitting, each component is matched using the Soundex™ algorithm. Missing elements will degrade the final score. Any remaining elements that are not fitted to one of the above fields are considered a "rest," and a score is computed via a coverage match. The coverage match is the average of two values, the fraction of elements in the first address string that are in the second string and the fraction of second string elements that are in the first. All of these distinct matches are weighted to yield the final address match. The following example shows the performance of the field address matching function. In the example, the field matching function matches address 1 and address 2 which are:

Address 1	Address 2
100 NW BIRK BARK RD PO BOX 22 11E	STE 11E 100 BIRCH BARK RD PO BOX 22

The field matching function uses the template fitting to compare the address attributes. The template fitting is as follows:

Template fitting:			Individual match
PO BOX:	22	22	soundex = 1
Street number:	100	100	soundex = 1
Street name:	BIRK BARK	BIRCH BARK }	
Street descriptor:	RD	RD } soundex = .91	
Street direction:	NW	— }	
Rest:	11E	STE 11E	coverage = .75

Using the aforementioned address rules, the actual address match score was 0.96.

The above field matching functions are then applied to the pairs of training data that were identified as similar to learn the plurality of matching rules. An example of the application of the field matching functions to the training data is shown in FIG. 5, which is a screen view 50 of the matching function. In FIG. 5, each record pair is displayed in the first two "white background" columns under the Record 1 and Record 2 headings and compared field by field, with the degree of match between fields listed under the match score heading. In this example, the business attribute (i.e., VFS), the country attribute (i.e., USA), and state attribute (i.e., CT) were an exact match. The name attribute had a matching score of 0.67 for MOMS SODA SHOP versus POPS SODA SHOP. The address attribute had a low matching score of 0.22 for 222 HOPEMEADOW STREET and 5 GRIFFIN ST. S. Other low matching scores occurred for the city attribute (SIMSBURY vs. BLOOMFIELD) and the Duns No. attribute. Relatively better matching scores occurred for the phone number and customer id attributes.

After all of the pairs of records have been evaluated, an intermediate file of vectors containing matching scores for all of the fields from each of the similar pair of records is generated. An example of an abbreviated intermediate file of vectors is shown below in Table 1.

TABLE 1

An Intermediate File of Vectors

name	addr	city	state	zip	cou	phon	cid	duns	uldu	ulna	sic	match
0.20	0.28	1.00	1.00	0.71	1.00	0.00	0.00	0.00	0.00	0.50	0.00	0
0.63	0.86	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.50	0.10	1
1.00	1.00	1.00	1.00	1.00	1.00	0.50	0.00	0.33	0.50	0.50	1.00	1
0.75	1.00	1.00	1.00	1.00	0.00	0.50	0.00	0.00	0.50	0.50	0.00	1
1.00	1.00	1.00	1.00	1.00	1.00	0.50	0.83	0.00	0.50	0.50	0.00	1
0.18	0.00	1.00	1.00	1.00	1.00	0.63	0.17	0.11	1.00	0.50	1.00	0

In each row there is a matching score between a pair of training records for each corresponding field. For example, in the first row, the matching scores between the name, address, city, state, zip code, country, phone, customer id, duns no., ult duns, ult name, and sic code fields are 0.20, 0.28, 1.00, 1.00, 0.71, 1.00, 0.00, 0.00, 0.00, 0.00, 0.50, and 0.00, respectively. The last column in the file designates whether the pair of records was a match (1) or a non-match (0).

The plurality of matching rules are then learned by converting the intermediate file of vectors into matching rules. Conversion of the intermediate file is achieved by using commercially available statistics applications. In the present invention, a decision tree induction algorithm called CART (Classification and Regression Trees) was used as part of an S-PLUS™ statistics package. These packages

generate an output that is in the form set forth below, where a line with an asterisk indicates a rule.

1)root 1142 223.100 0.266200

2) name<0.7 818 2.989 0.003667

4) phone<0.905 810 0.000 0.000000 *

5) phone>0.905 8 1.875 0.375000 *

3) name>0.7 324 21.370 0.929000

6) addr<0.625 23 0.000 0.000000 *

7) addr>0.625 301 0.000 1.000000 *

In each row there is an inequality, then the number of training cases, then a distance measure, and last a frequency of match. For example, line (4) indicates that when the name field was <0.7 and the phone field was <0.905, then there were 810 cases, and 0 percent of them matched. Arbitrary thresholds near 0 and 1 are then used to decide when two records should be called a "match," a "non-match," or a "possible match," In the case of a "possible match," it is best if an end-user is consulted for advice. The above output is then parsed into matching rules to be used for subsequent matching The parsed rules for the above output are as follows:

If name match >0.7 and address match >0.625 then definite match

If name match <0.7 and phone match >0.905 then maybe match—get expert approval.

Otherwise no match.

Additional rules can be subsequently added, as necessary, to this learned rule set, based on the user's domain knowledge and data quality. An example of an added rule is if name=1 and address=1 and phone=1, then definite match.

The learned matching rules can then be used for matching a new data record from the new records database **14**. Typically, a matching operation between a new customer data record and the candidate set will result in a list of zero to ten potential matches. If only one candidate is a definite match, then that record is considered to be the match for the new customer record. The existing customer database is then updated with data from the new customer record.

The existing customer record is updated by using the customer ID for the matched record. All fields of the existing record are then checked to see if the new record will provide additional information. If the existing record contains no information for a particular field and the new record contains information, then the new information is added. For addresses, information is added only if all existing address fields are blank or if the matching operation found new information like a PO Box where there was only a street address or vice verse. In any case, existing data is never changed.

If there are multiple "definite matches" or no "definite matches", but one or more "maybe matches", then those candidates are marked with a pending flag for entry into the pending file **22**. These candidates are then resolved at a later time.

If there are no matches between the new customer record and the existing customer records, then a new customer entry is made in the customer table of the existing records database **12**. Because the entry is a new record, a new unique customer ID has to be created for this particular customer record. The new customer database entry is then inserted with all available normalized data for this customer.

If the pending flag, described above, is true or there is no clear-cut winner among the list of potential matches, then the new customer data record does not go into the existing records database **12**. The entire new data record is written out to the pending file **22**, with those portions that have been normalized written to reflect the normalization. Also written is the list of customer IDs for the potential matches. This file can then be viewed later to determine which match, if any, is the correct match. FIG. **6** discloses a flow chart describing the process of examining pending data in the pending file for a match. The process begins by reading a data record from the pending file at **52**. The data record is then listed at **54**. Next, the correct potential customer record matches are selected by a user at **56**. The data record can be presented to the user on the left side of the display, with data from each of the potential customer record matches on the right side. At **58**, the user then decides what kind of matching should be done. For example, if the pending record is actually a new record, then the data is entered into a new record in the existing records database **12**. If there is a match with one or more records, then the user can mark the records and can select which one will get the updated information. Another option is to keep the pending data record in the pending file **22**. If there are more records as decided at **60**, then steps **52**, **54**, **56**, and **58** are repeated. Otherwise, the process ends if there are no more records.

It is therefore apparent that there has been provided in accordance with the present invention, a method and system for matching a new data set to an existing data set in a database that fully satisfy the aims and advantages and objectives hereinbefore set forth. The present invention has been described with reference to several embodiments, however, it will be appreciated that variations and modifications can be effected by a person of ordinary skill in the art without departing from the scope of the invention.

The invention claimed is:

1. The method for matching a new data set containing a record and a collection of fields to a database containing a plurality of records each having a collection of fields, the method comprising the steps of:

reading the new data set;

validating each of the fields from the record in the new data set;

normalizing the validated fields in the record in the new data set into a standard form;

selecting a hash key for generating a candidate set of records from the database that likely matches the record from the new data set;

applying the hash key to the plurality of records in the database to generate the candidate set of records;

matching the record from the new data set to each of the records in the candidate set; and

updating the plurality of records in the database according to the match of the record from the new data set to the records in the candidate set.

2. The method according to claim **1**, wherein the step of validating comprises:

using a plurality of validation tables containing standards for each of the fields;

comparing each of the fields from the record in the new data set to the standards; and

verifying that data in each of the fields from the record in the new data set correspond to the standards listed in the validation tables.

3. The method according to claim **1**, wherein the step of normalizing fields comprises removing excess punctuation and standardizing abbreviations therefrom.

4. The method according to claim **3**, further comprising the step of removing the record from further consideration if the fields are unsuitable for normalization.

5. The method according to claim 1, wherein the step of applying the hash key comprises searching each of the plurality of records in the database for records containing the selected hash key.

6. The method according to claim 1, wherein the step of matching comprises:

comparing the record from the new data set to each of the records in the candidate set; and

generating a matching score indicating a degree of match between the record from the new data set and each record in the candidate set.

7. The method according to claim 1, wherein the step of updating comprises adding the record from the new data set to the plurality of records in the database if there was a non-match.

8. The method according to claim 1, wherein the step of updating comprises editing the record in the database with the record of the new data for an exact match.

9. The method according to claim 1, wherein the step of updating comprises placing the record from the new data set in a pending file for a possible match.

10. A system for matching a new data set containing a record and a collection of fields to a database containing a plurality of records each having a collection of fields, the system comprising:

means for reading the new data set;

means for validating each of the fields from the record in the new data set;

means for normalizing the validated fields in the record in the new data set into a standard form;

means for selecting a hash key for generating a candidate set of records from the database that likely matches the record from the new data set;

means for applying the hash key to the plurality of records in the database to generate the candidate set of records;

means for matching the record from the new data set to each of the records in the candidate set; and

means for updating the plurality of records in the database according to the match of the record from the new data set to the records in the candidate set.

11. The system according to claim 10, wherein the validating means comprises:

means for using a plurality of validation tables containing standards for each of the fields;

means for comparing each of the fields from the record in the new data set to the standards; and

means for verifying that data in each of the fields from the record in the new data set correspond to the standards listed in the validation tables.

12. The system according to claim 10, wherein the normalizing means comprises means for removing excess punctuation and means for standardizing abbreviations.

13. The system according to claim 12, further comprising means for removing the record from further consideration if the fields are unsuitable for normalization.

14. The system according to claim 10, wherein the hash key applying means comprises each of the plurality of records in the database for records containing the selected hash key.

15. The system according to claim 10, wherein the matching means comprises:

means for comparing the record from the new data set to each of the records in the candidate set; and

means for generating a matching score indicating a degree of match between the record from the new data set and each record in the candidate set.

16. The system according to claim 10, wherein the updating means comprises means for adding the record from the new data set to the plurality of records in the database if there was a non-match.

17. The system according to claim 10, wherein the updating means comprises means for editing the record in the database with the record of the new data set for an exact match.

18. The system according to claim 10, wherein the updating means comprises means for placing the record from the new data set in a pending file for a possible match.

* * * * *

```
Set     Items   Description
S1   317410  VERIFY??? OR VERIFI?????? OR CHECK??? OR VALIDAT???
S2  1776670  COMPAR???? OR MATCH??? OR EQUAL??? OR UNEQUAL??? OR
             SYNCHRON???? OR UNSYNCHRON???? OR ASYNCHRON????
S3     5858  (DATABASE OR REFERENTIAL OR ENTITY OR KEY OR DOMAIN OR
                   COLUMN OR TABLE)(3N)INTEGRITY OR DATABASE(3N)(CHANGE OR
                   CHANGING OR CHANGED OR UPDAT??? OR ALTER??? OR MODIFY OR
                   MODIFI?????? OR EDIT??? OR VIOLAT??? OR CORRUPT??? OR
                   COMPROMIS???
S4  2176688  HASH??? OR MD5 OR CRC32 OR COMPRESS??? OR GZIP OR BZIP2
                   OR WINZIP OR ZIP OR REDUCED OR COMPACT OR MINIATURE OR
                   ABRIDGED OR SUMMARIZ???
S5  3304495  METADATA? ? OR META()DATA OR ATTRIBUTE? ? OR FIELD? ? OR
                   PROPERT??? OR DATAFOUNDRY OR UML OR SQL92 OR
                   DATA(2N)TYPE? ? OR SIZE OR LENGTH OR FILENAME? ? OR (FILE
                   OR RECORD)()NAME? ? OR PARAMETER? ? OR METAFILE? ?
S6        3  S1 AND S2 AND S3 AND S4 AND S5

 show files
  File 347:JAPIO Nov 1976-2005/Aug(Updated 051205)
         (c) 2005 JPO & JAPIO
  File 350:Derwent WPIX 1963-2006/UD,UM &UP=200606
         (c) 2006  Thomson Derwent
```

BIBLIOGRAPHIC
PATENT

4/3,K/14 (Item 14 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

00675968
Postal rating system with verifiable integrity.
Postgebuhrensystem mit nachprufbarer Unversehrtheit.
Systeme de tarification postale avec integrite de verification.
PATENT ASSIGNEE:
PITNEY BOWES, INC., (244953), World Headquarters One Elmcroft, Stamford Connecticut 06926-0700, (US), (applicant designated states: CH;DE;FR;GB;LI)
INVENTOR:
Pintsov, Leon A., 365 Mountain Road, W. Hartford, Connecticut 06107, (US)
Connell, Richard A., 24 Lower Salem Road, South Salem, New York, 10590, (US)
Sansone, Ronald P., 4 Trails End Road, Weston, Connecticut 06883, (US)
Schmidt, Alfred C., 201 Branch Brook Road, Wilton, Connecticut 06897, (US)
LEGAL REPRESENTATIVE:
Gorg, Klaus, Dipl.-Ing. et al (4311), Hoffmann, Eitle & Partner Patent- und Rechtsanwalte Postfach 81 04 20, 81904 Munchen, (DE)
PATENT (CC, No, Kind, Date): EP 647925 A2 950412 (Basic)
EP 647925 A3 951025
APPLICATION (CC, No, Date): EP 94307376 941007;
PRIORITY (CC, No, Date): US 133398 931008
DESIGNATED STATES: CH; DE; FR; GB; LI
INTERNATIONAL PATENT CLASS: G07B-017/04;
ABSTRACT WORD COUNT: 265

LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:

Available Text	Language	Update	Word Count
CLAIMS A	(English)	EPAB95	1894
SPEC A	(English)	EPAB95	8750
Total word count - document A			10644
Total word count - document B			0
Total word count - documents A + B			10644

...SPECIFICATION is publicly available data as to how mail should be rated for various different rating **parameters** . The rate table is communicated to the mailer along with a code. The code is...

...rate table is the appropriate sending authority. This both authenticates the source of the rate **table** and the **integrity** of the data received. In accordance with a feature of the embodiments printing by the...
...of the data stored within the postage evidencing device memory for the rate table is **verified** as being correct. This is done by recomputing the code for the rate table and...

...accordance with still a further feature of the embodiments the code (which may be a " **hash** " code) is printed along with the rating parameters on the mail piece such that a **verifying** party can reconstruct the rating process and determine if rating inaccuracy occurred and/or if...code may be printed along with other encrypted information on the mail piece. Alternatively the **hash** code may be combined with other information such as the postal value and postage evidencing...

...rating profile for a particular user or a group of users is stored by

the **verifying** party to enable the generation of a profile of a mailer or a group of...

...data for marketing to such mailer further postal services and/or informational reports based upon **verified** mailing patterns, such as rate, level of service, mail destination, distribution and the like.

Preferred...

...and in which:

FIGURE 1 is a mailing system employing a secure rating module allowing **verifiable** rating integrity;

FIGURE 2 is a flow chart of the activities of the data center...

...postal evidencing device involved with processing a received rate table and the process by which **verification** of the **integrity** of the rate **table** data and the authenticity of the data center is established in the postage evidencing device...another definition is that a hash function h is a function that satisfies the following **properties** : 1) it is capable of converting a file F of arbitrary length into a fixed...

...evidencing device 114. If the two hash values received from the data center and the **hash** value computed in the postage evidencing device match each other, the **integrity** of the rate **table** received and stored in the postage evidencing device rating module 116 is assured. Thus, the integrity of the stored rate table and/or calculation algorithm is **verified** .

Both steps (authentication of the data center and **verifying** the **integrity** of the rate **table** and/or calculation algorithm received) can be combined. To do so, the data center 112...

...calculation algorithm encrypted with the secret key. Thus, the authenticity of the sender and the **verification** of the message can be achieved in one step.

A description now follows in connection...

...to the postage evidencing device 114 at 214. Thereafter, the data center 112 computes the **hash** value (message digest) of the rate table at 216. The **hash** value is then encrypted by the data center 112 at 218. The encrypted **hash** value is transmitted to the postage evidencing device 114 at 220.

Reference is now made...

...postage evidencing device 114 at 322. The postage evidencing device 114 also receives the encrypted **hash** value of the rate table at 324. The postage evidencing device 114 then computes the **hash** value (message digest) of the received rate table and obtains a first **hash** value at 326. The postage evidencing device 114 decrypts the received encrypted **hash** value of the rate table at 328. This provides a second hash value at the...

...piece by printer 122 for detection at a mail piece verification facility. Several attempts to **verify** the integrity of the received rate table and/or calculation algorithm

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: 0 647 925 A2

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 94307376.7

(51) Int. Cl.[6]: G07B 17/04

(22) Date of filing: 07.10.94

(30) Priority: 08.10.93 US 133398

(43) Date of publication of application:
12.04.95 Bulletin 95/15

(84) Designated Contracting States:
CH DE FR GB LI

(71) Applicant: PITNEY BOWES, INC.
World Headquarters
One Elmcroft
Stamford Connecticut 06926-0700 (US)

(72) Inventor: Pintsov, Leon A.
365 Mountain Road
W. Hartford, Connecticut 06107 (US)
Inventor: Connell, Richard A.
24 Lower Salem Road
South Salem, New York, 10590 (US)
Inventor: Sansone, Ronald P.
4 Trails End Road
Weston, Connecticut 06883 (US)
Inventor: Schmidt, Alfred C.
201 Branch Brook Road
Wilton, Connecticut 06897 (US)

(74) Representative: Cook, Anthony John et al
D. YOUNG & CO.
21 New Fetter Lane
London EC4A 1DA (GB)

(54) Postal rating system with verifiable integrity.

(57) A data center provides a rate table to a user. The rate table is communicated to the mailer along with a hash code. The hash code is based on information from the rating table. The hash code provides a unique number based on the rating table provided. The algorithm within a secure device and to which the rate table is loaded regenerates the hash code based on the information received from the rate table and compares the transmitted hash code with the generated hash code. A comparison is made of the received hash code and the generated hash code to verify that the rate table data has not been intentionally or unintentionally corrupted. The transmitted hash code may be encrypted by the data center and when received decrypted by the mailer. The encryption decryption process establishes authenticity of the data center if desired.

The generation of a hash code based on the stored rate table and a comparison with a stored hash code previously transmitted can be initiated prior to postage printing and used to insure proper rating. Printing is enabled only after the rating process has been properly implemented. The hash code and rating information may be printed on the mail piece such that a verifying party can reconstruct the rating process and determine if rating inaccuracy occurred. Various rating inaccuracy for a particular user can be stored by the verifying party to detect a recurrence of rating errors. Rating profiles for particular users or group of users may be stored to enable generation of user profiles.

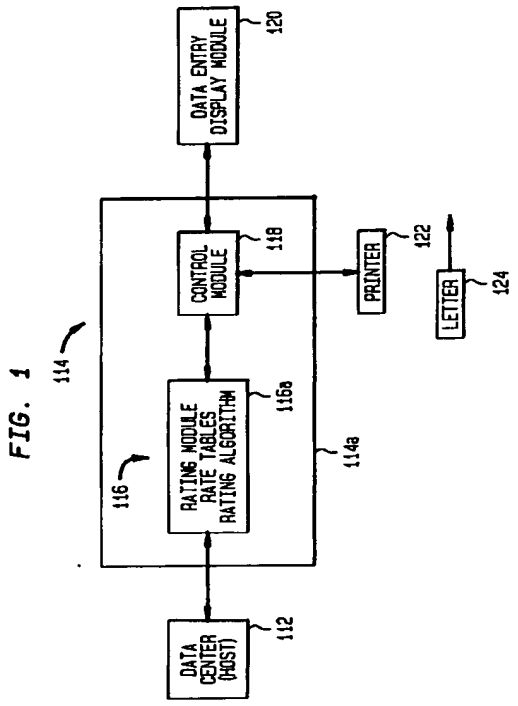


FIG. 1

EP 0 647 925 A2

The present invention pertains to rating of mail for postal systems, for example to a postal rating system having verifiable integrity determinable from the information printed on a mail piece.

Various postal services and private carrier services throughout the world have developed rate tables for mail and parcels. These rate tables specify the rate for any given mail piece (hereinafter intended to include parcels and other mailable items as well).

The rating may involve the desired class of service, such as first class or third class mail in the United States, the weight of the mail, the size of the mail, the distance of which the mail is to be sent, the level of service such as Express Mail involving delivery the next day, and/or a discount associated with a level of work sharing. Each postal service and each private carrier service usually establish their own rate tables for mail and parcels. Postal service as used herein is intended to apply equally to mean both governmental or other postal services and also private carrier services. Similarly, postal value as used herein is intended to apply equally to mean both governmental or other postal values and also private carrier service delivery charge and other values.

To facilitate a mailer applying proper postage or other charges (such as, for example, insurance or certified delivery or return receipt, etc.) to a mail piece or to a tape to be adhered to a mail piece, various devices have been provided such as scales which include rate tables to provide a visual indication to the user of the appropriate postage for the given mail piece to be deposited with the postal service. In some instances, these weighing devices having rating tables allow for the automatic setting of the postage meter print wheels wherein the scale includes a connection to an electronic postage meter and conveys setting information. This now enables a more rapid printing of postage and processing of the mail. One example of such a system is the Pitney Bowes PARAGON mailing system wherein mail is weighed and the postage meter print wheels automatically set for imprinting of the proper postage on a mail piece. Another system such as that disclosed in U.S. Patent No. 4,855,920 for POSTAGE ACCOUNTING DEVICE provides a secure accounting unit with a memory including a rate charge of postage rates for different classes of mail. Yet another system is disclosed in U.S. Patent No. 5,191,533 for FRANKING MACHINE wherein rate tables are stored in a meter and are employed to set the printing mechanism to a desired amount.

It has been recognized that a mail piece may be imprinted with an improper postage amount. This can be due to a number of different factors such as the utilization of a wrong rate table, the utilization of an obsolete rate table, or the input of inaccurate data for the rating process. One example would be the input of an incorrect size of the mail piece (where the size of the mail piece is a rating factor).

Summary of the Preferred Embodiments of the Invention

It has been discovered that a rating system can be provided which allows verification of the integrity of the rating process.

It has further been discovered that it is possible to allow verification in a manner which determines that an appropriate rate table has been employed and to identify the reason for improper rating of the mail.

The embodiments facilitate the entry of rate tables (or their equivalent) into a postage evidencing system such as a postage meter, so as to increase the security of mail rating and provide assistance in determining that a mail piece was securely rated and that the right rate table was used in the rating process.

In accordance with the embodiments a data center (which may be run by a third party or by the postal service) provides a rate table to a user. The rate table is publicly available data as to how mail should be rated for various different rating parameters. The rate table is communicated to the mailer along with a code. The code is based on information from the rate table. The code provides a unique number based on the rating table provided. The algorithm within a secure device into which the rate table is loaded regenerates the code based on the information from the received rate table and compares the transmitted code with the generated code. The comparison results in an appropriate match if the rate table is authentic and if the source of the rate table is the appropriate sending authority. This both authenticates the source of the rate table and the integrity of the data received.

In accordance with a feature of the embodiments printing by the postage evidencing device, such as a postage meter, is not enabled until the integrity of the data stored within the postage evidencing device memory for the rate table is verified as being correct. This is done by recomputing the code for the rate table and comparing the code for the rate table with a stored code received from the data center when the table was originally transmitted which has been stored in a non-volatile memory. If the two codes are the same, printing is authorized.

In accordance with still a further feature of the embodiments the code (which may be a "hash" code) is printed along with the rating parameters on the mail piece such that a verifying party can reconstruct the rating process and determine if rating inaccuracy occurred and/or if the rate table employed in rating is valid for the

date of the postage imprint. The code may be printed in encrypted form on the mail piece and the encrypted code may be printed along with other encrypted information on the mail piece. Alternatively the hash code may be combined with other information such as the postal value and postage evidencing device identification and the combined result then encrypted and printed on the mail piece.

In accordance with yet another feature of the embodiments the rating inaccuracies for a particular user can be stored by the verifying party to detect a recurrence of rating errors and to automatically initiate appropriate corrective and/or other actions should, for any given mailer or group of mailers, rating errors of particular categories exceed certain threshold levels.

In accordance with still another feature of the embodiments the rating profile for a particular user or a group of users is stored by the verifying party to enable the generation of a profile of a mailer or a group of mailers to provide business data for marketing to such mailer further postal services and/or informational reports based upon verified mailing patterns, such as rate, level of service, mail destination, distribution and the like.

Preferred embodiments of the present invention will now be described with reference to the following figures wherein like reference numerals designate similar elements in the various views and in which:

FIGURE 1 is a mailing system employing a secure rating module allowing verifiable rating integrity;

FIGURE 2 is a flow chart of the activities of the data center involved with transmitting to a secure rating module a rate table in accordance with the present invention;

FIGURE 3 are the activities at the postal evidencing device involved with processing a received rate table and the process by which verification of the integrity of the rate table data and the authenticity of the data center is established in the postage evidencing device;

FIGURE 4 is a flow chart within the postage evidencing device for rating a mail piece and printing the appropriate Postal Revenue Block on the mail piece;

FIGURE 5 is a flow chart of a sub routine within the Authenticate Rate Table and Rate Computation Algorithm block of FIGURE 4; and,

FIGURE 6 is an imprint on a mail piece in accordance with the present invention.

General Overview

The postage value (rate) for every mail piece may be encrypted together with other data to generate a digital token. A digital token is encrypted information that helps to authenticate the value or other information imprinted or to be imprinted on a mail piece. Examples of systems for generating and using digital tokens are described in U.S. Patent No. 4,757,537 for SYSTEM FOR DETECTING UNACCOUNTED FOR PRINTING IN A VALUE PRINTING SYSTEM; U.S. Patent No. 4,831,555 for UNSECURED POSTAGE APPLYING SYSTEM; and U.S. Patent No. 4,775,246 for SYSTEM FOR DETECTING UNACCOUNTED FOR PRINTING IN A VALUE PRINTING SYSTEM. The entire disclosure of these three patents is hereby incorporated herein by reference.

As a result of the digital token incorporating encrypted postage value, altering of the printed postage value in a postal value revenue block is detectable by a standard verification procedure. Thus, to underpay postage, an attempt may be made to interfere with the rating process (as opposed to the resulting printed postage value).

Rating with verifiable integrity in accordance with the system described herein helps to: 1) provide diagnostics to the party conducting verification to enable detection of inadvertent misrating of mail pieces; and 2) provide evidence to the party conducting verification of deliberate underrating of mail pieces.

Rating input parameters may be entered into a system manually or automatically or partially manually and partially automatically. For example, sensory data such as weight, size of mail pieces and presence of a barcode can be automatically entered while desired level of service or mail class can be keyed in manually or entered by default from a file. Alternatively all rating parameters can be entered into the system manually. The process of computing the postal value (or rate) is based on calculations involving input rating parameters and a rate table. The process of mail rating, however, can produce incorrect results. The following are such examples:

A) Entered incorrect rating parameter or parameters (e.g. wrong entered weight or size).

B) The rate table is obsolete or the wrong rate table.

C) The rate table is incorrect because it has been deliberately altered.

D) Entered input rating parameter or parameters are incorrect and the rate table is obsolete or incorrect.

E) Entered input rating parameter or parameters are incorrect and the rate table has been deliberately altered.

It should, of course, be recognized that the above examples can be combined to produce additional examples such as A and B or A and C or B and C or A and B and C.

The case of inadvertent misrating can occur due to incorrectly entered data, or obsolete or incorrect rate table or both. In the above examples, the case of inadvertent misrating is equivalent to examples A, B or D. In

this case printing values of rating input parameters and rate table identification in the postal revenue block (or other area) on a mail piece provides required diagnostic data for a verifying party. Upon entering values of input parameters and rate table identification from the postal revenue block into a computer, the verification party is able to reproduce the rating process that took place during mail rating by the mailer. The verification party is also able to independently determine correct rating parameters and compute a correct rate. If the two rates obtained do not match, a pairwise comparison of rating parameters and rate table identification helps provide the desired diagnostic as to the reason for the misrating of the mail piece. In this manner, detecting the deliberate entering of incorrect rating parameters is also facilitated.

Examples C and E are cases of deliberate underrating. For the purpose of providing evidence of deliberate underrating it is desirable to help establish that the rate computation was altered by changing of the rate table or using a wrong rate table. In the case of example C or E, a user of a postage evidencing device might attempt to change certain memory locations where particular postal rates are stored. This can be prevented by using well known techniques such as a non-volatile memory (NVM) within a secure postage evidencing device housing for the storage of the rate table as it was just described. The secure housing is both resistive to tampering such as by the use of break off screws and also may provide forensic evidence of the fact of tampering. If this process is too expensive, especially for large rate tables or where regular updates of NVM in a secure manner are proved to be unacceptably more expensive than updates of regular type memory, a modification of present invention described below can be applied that detects the alteration of a rate table rather than preventing it. From the security point of view, the ability to detect the reason for misrating serves as an excellent deterrent measure since the reason for misrating can be proven and also since deliberate misrating of mail may constitute a criminal offense.

The rate table may be loaded into the RAM memory of a postage evidencing device (rather than a secure non-volatile memory) from a data center as is shown in FIGURE 2 and FIGURE 3 (which are described in detail hereinafter). The process insures the integrity of the rate table received from the data center by the postage evidencing device.

Another way to provide verifiable integrity of the mail rating process is to compute the hash value of the entire rate table (or its specified portion) upon each access to the rate table. Immediately after this hash value has been computed it is sent to a private (secure), non-volatile memory. This private memory can be accessed only by the encryption module of the postage evidencing device. This encryption module encrypts the hash value of the rate table actually used for rating, together with other information, into digital tokens. In other words this hash value serves as one of the elements of the postal data used by the digital token transformation to produce the encrypted information to be printed on a mail piece. The overall operation provides a digital signature of the rate table actually used by employing techniques known in modern cryptology (see for example Contemporary Cryptology, The Science of Information Integrity, ed. G. Simmons, IEEE Press, 1992).

Yet another way to detect deliberate alteration of the rate table is to use a function such as a hash function parameterized by a secret key. In this case, just as in the previously described case, the hash value of the entire rate table (or a suitable portion thereof) is computed after each access to the rate table. The hash value in this case is a function of a secret key and thus can not be computed without knowledge of this key. When the hash value is computed a small or truncated portion of it can be printed in the postal revenue block as a rate table identification. Typically, two decimal digits would be sufficient (since it gives a potential adversary only 1 chance out of 100 to guess the right value of the rate table identification.) These two decimal digits would appear completely random to any observer without knowledge of the secret key. These two digits (or any larger number of such digits) may be termed the rate table digital token. It may be a part of the digital token previously described. The hash function parameterized by a secret key can be computed as a Message Authentication Code (MAC) which is widely used in the financial services industry.

Detailed Description of the Preferred Embodiments

Reference is now made to FIGURE 1. A data center 112 contains various rate tables published by a postal service or other carrier. The rate tables provide the delivery charges or postal fees for various types of services depending on the various parameters for each category of service. For example, a rate table may exist for the United States Postal Service first class mail, providing rates for first class mail, depending upon the different weights associated with the mail piece. In contrast, rates for a parcel may include the ZIP code or zone code as part of the rating parameters to determine the appropriate fee or payment for delivery of such parcel. These rate tables are communicated, as for example by modem, by disk, by magnetic or smart card or by other suitable means, to a postage evidencing device shown generally at 114. The postage evidencing device may be a traditional electronic postage meter such as disclosed in U.S. Patent No. 4,675,841 for MICROCOMPUTERIZED ELECTRONIC POSTAGE METER SYSTEM; U.S. Patent No. 4,301,507 for ELECTRONIC POSTAGE METER

HAVING PLURAL COMPUTING SYSTEMS; other types of metering system for evidencing postage such as, for example, as disclosed in U.S. Patent No. 4,757,537 for SYSTEM FOR DETECTING UNACCOUNTED FOR PRINTING IN A VALUE PRINTING SYSTEM; or U.S. Patent No. 4,934,846 for FRANKING SYSTEM. The postage evidencing device (which may be a personal computer type metering system, however) should preferably have the ability to print variable information on a mail piece to provide the requisite information for verification by a verifying authority as will be hereinafter explained.

The postage evidencing device 114 includes a rating module 116. The rating module stores the rate tables which are communicated to the the postage evidencing device from the data center 112. The rating module 116 is operatively connected to a control module 118 which would include a central processing unit and various other suitable electronic components and program control devices such as programmable read only memories (PROMs), random access memories (RAMs) and non-volatile memories (NVMs) for storing various postal and accounting data. Many system architectures are suitable for the present invention. For example, the accounting circuitry and NVM(s) can be part of the rating module within the secure housing 116a (tamper resistant device housing) or within a separate secure housing. The housing 114a may be a secure housing, or distributed processing systems may be employed.

A data entry module 120 is provided to allow a user to enter information into the postage evidencing device 114. This data may include, for example, the weight, size, class of service and other data concerning the mail piece and relevant to the rating and mail finishing processes. Examples of the types of data that can be entered by a user includes mail class, weight, dimension (length, width, or thickness or all of them), desired service level, work share level (for the United States Postal Service these may include indication of due presence of certain bar code, ZIP code, or ZIP + 4 code, ZONE code or presort level, etc.). Yet another type of data that could be entered could be, for example, a graphics code for the graphics to be printed. It should be recognized that any other factors that are deemed to be relevant by a particular postal service carrier in the rating process may be enterable by the user through the data entry module 120. The entry can be manual or automatic; the data may be from a computer system associated with creating or tracking the mail pieces or it may be scanned or measured from the mail piece itself. A printer 122 such as a thermal printer or ink jet printer or pin printer or laser printer is coupled to the control module.

It should be recognized that the rating process can be viewed as mapping of a set of input parameters (which can be called a vector) into a set of rational numbers which represents the postal rates. This can be viewed as mapping f from a set of input vectors {I} into a set of numbers R which represents the postal rates. As an example, the input vector (that can consist of such components as: a) two ounce weight category, b) zone three, and c) a size indicator) can be mapped into a unique and specific rate, for example, 43 cents. As each of the vector components change, the rate changes. If the size indicator is eliminated and the mail piece was not, for example, oversized, the rate, for example, could diminish to a lower rate. A further example would be a one ounce letter with no zone category and no oversize category and no presort or other worksharing that would yield still a different rate. Thus, the various vectors (rating parameters) which constitute the input for the rate table determine the rate. As vectors change the rates may go up or down depending on the particular rate table involved. These parameters for rating vary from postal service to postal service and carrier to carrier. The rating parameters can be any number of parameters depending applicable rating criteria. These rating parameters will lead ultimately to a single price that is to be paid as determined by the appropriate rate table. Thus, input "vectors" can be utilized as the rate table input to map onto the rate table in the postage evidencing device or system rating module to establish the actual postage to be imprinted on the mail piece. It should be specifically recognized that the establishing of the postal value to be imprinted on a mail piece may require the utilization of more than one rate table. For example, a rate table may exist for delivery charges, and a separate rate table for mail piece insurance charges.

Another explanation of how the rating process can be viewed as a mapping from a set of input vectors {I} into a set of numbers R which represent postal rates is as follows: An input vector is an ordered set of numerical parameters:

$$I = (a_1\ a_{2,}\ ...,\ a_n)$$

where

a_1 is the weight of the mail piece,

a_2 is the length of the mail piece,

a_3 is the width of the mail piece,

a_4 is the thickness of the mail piece,

a_5 is the desired level of service (including delivery time, special processing request such as registered mail etc.)

a_6 is a postal code of the origination address

a_7 is a postal code of the destination address

a_8... a_n are other relevant parameters including the level of worksharing (presort, prebarcoding etc.)

Again, the parameters of I form an exhaustive set in a sense that it can include all relevant parameters for any postal system in any country.

The mapping from {I} into R is defined by the process of computing a rating function. This can be either an algorithmic computation or (the most common case) a special algorithm called the table look up wherein a pointer is generated that points to the particular code in a look up table for the rate based on the input vector.

The integrity of the rating process involves the integrity of computing the rate for any given mail piece. That is, for example, the integrity in employing the computational algorithm such as the integrity in utilizing a look up table. Integrity or the rating process requires the use of securely correct rates.

In one embodiment the computational algorithm itself and/or the rate table are encrypted by using a secret or public key encryption system transmitted to the rating module 116 of the postage evidencing device 114. The decryption algorithm can be initiated upon receiving a secret key or other private information by the rating module 116. The transmission can be accomplished via a modem in a traditional way known to those skilled in the art, or by direct phone contact with a user and hand data entry. Additionally, of course, all of the previously noted communication techniques for transmitting data can be employed. In the case of a stand alone system, not involving a data center, the decryption key must be stored in a physically protected memory location in the postage evidencing device (e.g. in the rating module). The encryption and decryption can be by any number of well known encryption/decryption techniques such as the Data Encryption Standard (DES) or the RSA system.

Upon receiving the rate table(s) and calculation (computation) algorithm, and decrypting them, if necessary, the verification of the rate table authenticity can be made as will hereinafter be explained. The calculation algorithm and rate table are stored in a protected data memory such as in a secure non-volatile memory. Both the rate table and the calculation algorithm have unique identifiers. The identifiers can be in the form of a code which also may include data indicative of the date of issue and/or the end date (time period) after which the calculation algorithm and rate table can no longer be considered valid. Additionally, data concerning the source of the data itself (the data center from which the data came) may also be included.

The task of mail rating can be accomplished in the following manner. First, the operator of the postage evidencing device (e.g., postage meter or shipping or weighing system) enters the input parameter $I = (a_1, a_2, \ldots a_n)$ of the mail piece to be processed. Alternatively, an automatic device (such as a mailing machine) can automatically measure some or all of the components of the vector I and enter it into the rating module; other components can be prestored and used as default parameters. In either case, the rating module performs a consistency check of the vector I in order to determine that the vector I can serve as a legitimate input for the rating process. Thus, all of the input parameters for rating are verified to check their legitimacy or logical consistency given the rating system being employed. (For example, entering a weight of three pounds for a letter class mail piece would not pass the test of consistency in the United States.) Then, the supervisory routine of the rating module invokes the rating algorithm and the rate table. This is done using one of the techniques well known in the art such as authentication channels, e.g. symmetric or asymmetric cipher exchange (see, for example, a book entitled Contemporary Cryptology, ed. G. Simmons, IEEE Press 1992). After the rate R is calculated the following data elements are passed to the postal rating revenue block formatting module (here the indicia or imprint is defined as a printed image that is to be used for evidencing postage payment). This can include rates (in the appropriate units of currency), identification of the rate table, identification of the rate calculation algorithm, and the rate input vector $I = (a_1, a_2, \ldots a_n)$. Some or all of these items of information are printed by the printer 122 on the mail piece 124 to enable verification. One verification approach involving video recording of mail pieces for later processing is disclosed in U.S. Patent Application of Robert A. Cordery and Leon A. Pinstov, Application Serial No. 08/077,667, filed June 18, 1993 for MAIL PROCESSING SYSTEM INCLUDING OFF-LINE VERIFICATION (equivalent to European Patent Application No. 94304236.6).

The postal revenue block (indicia) formatting module combines these data elements with others (such as, for example user device identification, date/time stamp, postal codes of origination and destination, and possibly others, as for example suggested in the above-identified three U.S. patents which have been incorporated herein by reference or also in U.S. Patent No. 4,853,961 for RELIABLE DOCUMENT AUTHENTICATION SYSTEM, the entire disclosure of which is also hereby incorporated herein by reference. This generates a printable digital image of the postal revenue block.

The authentication channel for rate table communications between the data center 112 and the postage evidencing device 114 will now be described. The authentication channel is well known in the art (see for example Contemporary Cryptology ed by G. Simmons, IEEE Press, 1992). The authentication channel involves two communicating parties who would like to authenticate each other before exchanging any sensitive messages. The parties can be a data center and a postage evidencing device.

The data center would operate to send a rate table to a postage evidencing device via a communications channel (phone line or other transmission). The secret information (for example, a secret key in a case of a secret key based protocol) is stored both at the data center and in the postage evidencing device. Alternatively, in a public key system one of the parties (for example, the data center) knows a secret key, and the other party (here, the postage evidencing device) knows a matching public key. The protocol for mutual authentication requires that the data center first sends information in plain text and then the same information encrypted with its secret key. The postage evidencing device upon receipt of both messages deciphers the encrypted message with its secret (or public)key and compares it with its plain text version. If a match is made, the data sender is authenticated, since only the sender knew the secret key. Similarly, the postage evidencing device can send two messages, plain text and encrypted message to authenticate itself to the data center if needed. In mailing applications this may not be needed.

After such authentication, if it is desired, the data center 112 transmits a rate table and/or calculation algorithm. This transmission, however, requires a data integrity. That is, that the rate table and/or calculation algorithm should arrive unmodified. Assurance is needed that the rate table and/or calculation algorithm arrives exactly as it was sent and that it has not been corrupted, intentional or unintentionally. In order to accomplish this, the data center 112 first generates a hash value (message digest) of all or some specified portion of the data contained in the rate table and/or of the calculation algorithm to be sent. The rate table and/or calculation algorithm can then be sent as an ASCII or other type of file. The hash function applied to this data produces a hash value (message digest) which is indicative of the content of the rate table and/or calculation algorithm and yet is considerably reduced in data size. As used herein hash function is a well known function which possesses at least two properties. It is computationally difficult to (i) recover a message corresponding to a given message digest and (ii) to find two different messages which produce the same hash value (message digest). Some well known hash functions are described in American National Standard X9.30 - 1993, Public Key Cryptography Using Irreversible Algorithms For The Financial Services Industry: Part 2: The Secure Hash Algorithm (SHA). It should be noted that there are other publicly available hash functions that can be implemented for the purpose of the present invention. As for example, one formal definition is set forth in Contemporary Cryptology by G. Simmons, IEEE Press 1992 at page 345, and yet another definition is that a hash function h is a function that satisfies the following properties: 1) it is capable of converting a file F of arbitrary length into a fixed-length digest h (F); 2) h must be "one way", that is, given an arbitrary value y in the domain of h, it must be computationally infeasible to find file F such that h (F) = y; and 3) h must be "collision free", that is, it must be computationally infeasible to construct two different files F_1 and F_2 such that h (F_1,) = h (F_2).

Since the data (the rate table and/or calculation algorithm) being transmitted to the postage evidencing device 112 is publicly available information, it is not necessary to encrypt the information and prevent unauthorized decryption since it is not important to protect secrecy of the information itself. Upon calculation of the hash value (message digest) of the rate table and/or the calculation algorithm the data center encrypts the hash value (message digest) with its secret key (for both public and secret key systems) and sends the encrypted message to the postage evidencing device 114. The postage evidencing device 114 receives the encrypted hash value ("signature"), and decrypts it with its secret or public key as the case may be, thus obtaining the plaintext hash value (message digest). The postage evidencing device 114 then independently computes the hash value (message digest) of the received rate table and/or calculation algorithm using the same hash function. The hash algorithm employed may be one in the public domain; however the algorithm resides both at the data center 112 and at the postage evidencing device 114. If the two hash values received from the data center and the hash value computed in the postage evidencing device match each other, the integrity of the rate table received and stored in the postage evidencing device rating module 116 is assured. Thus, the integrity of the stored rate table and/or calculation algorithm is verified.

Both steps (authentication of the data center and verifying the integrity of the rate table and/or calculation algorithm received) can be combined. To do so, the data center 112 simply sends two messages to the postage evidencing device 114: the rate table and/or calculation algorithm in plain text and the rate table and/or calculation algorithm encrypted with the secret key. Thus, the authenticity of the sender and the verification of the message can be achieved in one step.

A description now follows in connection with FIGURES 2, 3 and 4 of the activities of a rate table/calculation algorithm at the data center 112 and at the postage evidencing device 114.

Reference is now made to FIGURE 2. The data center 112 sends the rate table and/or calculation algorithm to the postage evidencing device 114 at 214. Thereafter, the data center 112 computes the hash value (message digest) of the rate table at 216. The hash value is then encrypted by the data center 112 at 218. The encrypted hash value is transmitted to the postage evidencing device 114 at 220.

Reference is now made to FIGURE 3. The rate table is received by the postage evidencing device 114 at 322. The postage evidencing device 114 also receives the encrypted hash value of the rate table at 324. The

postage evidencing device 114 then computes the hash value (message digest) of the received rate table and obtains a first hash value at 326. The postage evidencing device 114 decrypts the received encrypted hash value of the rate table at 328. This provides a second hash value at the postage evidencing device 114.

A comparison is made at 330 of the first hash value which has been computed by the postage evidencing device 114 and the second hash value which has been obtained by decryption. If a match is made at 322, the process continues at 334 and may ultimately result, when required, in printing of the postal revenue block. This would occur if all other conditions are appropriate in the postal evidencing device, as for example adequate funds are available for postage printing. If a match has not been made at 332 the process is stopped at 336, since the integrity of the received rate table and/or calculation algorithm has not been verified. The postage evidencing device 114 may be inhibited from further operation, if desired, requiring physical inspection and servicing. Alternatively the system may be allowed to operate but an error flag set in the postage evidencing device 114 and printed on a mail piece by printer 122 for detection at a mail piece verification facility. Several attempts to verify the integrity of the received rate table and/or calculation algorithm may be allowed before the postage evidencing device is locked up.

The value of the hash function (or a part thereof) can serve as a unique rate table identification number. This unique identification number can be associated with the validity period of the rate table in a one to one fashion. For example, the rating authority (the postal service or other carrier) provides identification for each new rate table and creates a table where both information as to the rate table identification and corresponding validity periods are stored. A simple table look up allows the verifying facility, mailer or third party to recover the validity period. This is useful for the postage payment verification process. In this instance by utilizing the unique identification number (as for example a hash value) the verification service can determine the specific postal or carrier rating table utilized and thus can determine whether the rating table used by the mailer in calculating the mail piece rate and thus postage value imprinted on the mail piece was within the validity period. Moreover, it should be expressly recognized that it may be desirable to encrypt the printed hash function or have the hash value parameterized by a secret key. Thus the printed encrypted or parameterized value of the hash function on the mail piece is not subject to attack and can itself be verified. This technique of imprinting an encrypted or parameterized hash value on the mail piece can be employed with each of the various aspects and embodiments of the present invention.

Enhanced verifiable integrity of the rate computation itself is also provided by the present system. There are a number of ways that the system can compute rates with verifiable integrity. Depending upon the particular implementation, there can be different systems requirements, as for example the speed of the processor and the storage capabilities of the RAMs and NVMs.

One way to achieve this enhancement of the integrity of the mail rating process is to load the rate table (as previously described) together with its identification into the non-volatile memory of the rating module 114. The system requires access to and use of the rating table and/or calculation algorithm before enabling printing of the postal revenue block (meter indicia). This may be accomplished, for example, by precluding access to the postal revenue block formatting software module until the rating vectors have been entered and the rating process completed. Another manner in which this can be accomplished is to load the rate table and/or calculation algorithm together with its unique identification into the non-volatile memory of the rating module 116. The postage evidencing device 114 central control program operates such that only access to this non-volatile memory and the appropriate rating process memory locations therein can trigger printing of the postal revenue block (meter indicia). Postal value thus cannot be printed without access to the rate table and/or calculation algorithm.

Another way to provide enhanced (verifiable) integrity of the mail rating process is that, upon entering required rating input parameters, the postal evidencing device 114 invokes a control routine which computes a pointer to the rate table for a given mail piece. This can be done by formatting the rate table first as a multi entry numeric table or multidimensional array having a number of dimensions equal to the number of input parameters. The pointer can be a concatenated string of numbers or symbols partitioned into sections indicative of the appropriate location in the array. The number of sections is equal to the number of input parameters.

For example, if the rate table has only three weights, 1, 2 and 3 ounces, two dimensional indicators (zero being indication of regular size and one being indicative of oversized mail piece) and two delivery service classes, 0 (delivery within three days from the moment of deposit) and 1 (delivery within six days), then the pointer may be the number 201. This would mean that mail piece weighting 2 ounce, having regular size and scheduled for delivery within 6 days needs to be rated. The pointer points to only one corresponding rate in the table for such rating e.g. 43 cents. This rate can be retrieved after a hash value for the entire table or its specified portion has been computed and compared with hash value (message digest) for the table or its specified portion received from the data center 112 and stored in the non-volatile memory 114 of the postage evidencing device. This approach reduces the size of the required non-volatile memory needed to store rate table information. If

the hash values (message digest) match, verification is established, which means that an uncorrupted rate table was used for the rating process. The rate value together with the rate table identification are retrieved and sent to a postal revenue block formatting routine for formatting the data for printing.

The flow chart in FIGURE 4 shows the activities in the postage evidencing device 114 for rating a mail piece and printing the appropriate postage payment on the mail piece 124.

Reference is now made to FIGURE 4. A user enters rating parameters into the postage evidencing device 114 at 438. The postage evidencing device 114 verifies the consistency of the mail piece parameters at 440. A verification message is then sent at 442. If consistency has not been established at 443, the mail piece is rejected at 445. If consistency has been established at 443, the rate is computed at 444.

As part of computing the rate, the rate table and rate table calculation (computation) algorithm are authenticated at 446. An authentication message is sent at 448. If authentication has not been established at 450, the rate table is rejected at 452 and the process is not allowed to proceed. Thus, the rate computation noted above will not occur. If the authenticity of the rate table has been established at 450, the computation at 444 is enabled based on the authenticated rate table and on the verified mail piece parameters. The computed rate is sent to the postage printing formatting module at 447.

Reference is now made to Figure 5. The activities within the postage evidencing device 114 relating to authenticating the rate table as shown in Figure 4, block 444 involves a series of steps. Initially, after receiving the verification message of consistency of the mail piece parameters, a pointer is computed to the rate table based on the parameters at 544. The hash value (message digest) of the rate table is computed at 546. The computed hash value (message digest) of the rate table is compared with the hash value (message digest) of the rate table stored in the postage evidencing device non-volatile memory at 548. If the hash values do not match at 548, the process is stopped at 549 and various alternatives can be implemented as previously noted including locking up the postage evidencing device, allowing the number of lead tries or setting a flag in the postage evidencing device NVM.

If the hash values (message digest) match at 548, access to the rate table itself is enabled at 550 and the rate involved is obtained. The rate is formatted as part of the revenue block enabling the postage evidencing device to be prepared to print at 552. The postage evidencing device printer 122 is then enabled for printing at 554 and printed at 556. The formatting of the postal revenue block will include the hash value (message digest) as well as the rate to enable later identification. All or a part of the information contained in the hash value can be utilized to determine the authenticity, validity, and currency of the rate table. Moreover, the rating vectors (rating parameters) are also printed. As previously noted the hash value may be encrypted or parameterized by a secret key. This prevents the use, for example, of improper rating vectors or rate table and the deliberate altering of the hash value or part thereof for the proper rating vectors and proper rate table.

Reference is now made to Figure 6 which is a representative mail piece with one example of the type of information which may be printed on the mail piece 124. It should be recognized that the printed information and its organization are a matter of choice and can be printed at different locations on the envelope panel or tape; moreover, the information relative to a mail piece may be stored with a mail piece and/or mailer identifier code for later processing and analysis. The stored data for later analysis can be for a single mailer or a group of mailers. The data will provide information concerning mailing patterns and information regarding rating experience for any such mailer or group of mailers.

The formatted printed postal revenue block in the present example includes a postage evidencing device identification number 612, a town circle 614, and a postage amount and suitable indicia design which may include graphics of which could change with the value and the amount 616.

Printed at the bottom of the postage printing block 600 is a sequence of information segments including the hash value or part thereof (message digest of the rate table and/or calculation algorithm 618). As noted this hash value may be encrypted or parameterized. This value provides identification of the rate table itself and/or calculation algorithm, as previously described. The weight classification of the mail piece is printed at 620 and the desired level of service is printed at 622 (one day delivery, three day delivery, 6 day delivery, etc.). The class of service, for example, registered mail,is printed at 624 and a flag for oversized mail piece is printed at 626. A workshare level such as presort, barcoding, etc., is printed at 628.
To facilitate rapid scanning of the printed information a barcode representation of some or all of the information previously noted is printed at 630.

It should be clearly recognized that the information printed, its location, the fonts used, the bar code types and styles are all a matter of design choice and can be modified to meet the needs and requirements of the particular postal service or private carrier or mailer involved, depending upon the conventions established for these matters. Moreover, the problem of checking of stores and retrieves from a memory such as a RAM is known in the art (see for example Checking The Correctness of Memory, by M. Blum, et al, Proceedings of the IEEE Symposium Foundations of Computer Science, Pages 90-99, 1991).

The following is an example of some of the aspects of the above described systems:

The example utilizes the technique described in Contemporary Cryptology, ed. G. Simmons, IEEE Press, 1992, on page 392).

The first 64 bits of the rate table (or its suitable portion) are block-encrypted using DES and a secret key. Then the next 64 bits are added to the just produced cipher. The result is block-encrypted again using the same key, producing a new 64 bits of cipher. The procedure continues until all the 64 bits blocks of the rate table have been processed. The technique of padding with zeroes the last block (which is typically less than 64 bits) is applied.

Consider the following example. A portion of the current United States Postal Service rate table for letter mail weighing under one ounce can be represented as a string of numbers, namely: RT = 110290 120267 130248 140230 150242 160239 170233.

Here each 6 digit segment represents one rate which is a function of weight, encoding, presort and prebarcoding attributes. There are total 7 possible combinations of these attributes, i.e. currently the mail could be presorted to 3 or 5 digit levels, prebarcoded or ZIP+4 numerically encoded. For example, the combination number 6 implies that the mail is presorted to 3 digit level and prebarcoded by mailer. This type of mail weighing less than 1 oz. should be postaged at $0.239 per piece. This corresponds to the segment of rate table (160239) where the first digit 1 is indicative of the weight under 1 oz., the second digit 6 is indicative of the above explained combination of encoding, presort and prebarcoding attributes and digits 0239 represent the postal rate itself.

A secure hash value of the rate table RT is generated. See Appendix A with the actual calculations. The hash value of the rate table is

6825965425726402962

The last two digits 62 of the hash value represent the rate table digital token.

The above described example allows one to reliably detect any attempt to substitute a correct rate stored in the rate table by a lower value. Thus, the deliberate alteration of the rate table described in example C can be detected and a printed evidence of such alteration can be provided to the verification party. It should be recognized that the other encryption techniques are suitable for use with the present invention. One such example is described in a paper by M. Blum et al, "Checking the correctness of Memories", Proceedings of 31st Symposium on Foundations of Computer Science, October 1990.

It should be recognized that the above described systems enable a postal service or other party to verify, authenticate and reproduce the rating process from the information imprinted on the mail piece. This allows auditing to insure that the rating process used to establish the rate was accurately and properly implemented. This includes that the correct rate table was used, consistent mail piece parameters or vectors were used as printed, the correct calculation algorithm was used, and the correct postage value was imprinted on the mail piece. The hash value (message digest) verifies that the correct rate table/calculation algorithm was used for rating, and with different vectors (parameters) such as the desired service, weight, etc., also imprinted on the mail piece, the rating process can be reconstructed. Moreover, the entire hash value 682596542572640962 (which has been parameterized with a secret key) results in the digital token being 62. This digital token 62 can be printed on the mail piece for verification purposes.

The present system thus enables an audit for each mail piece. The audit may not only determine if the mail piece was correctly or incorrectly rated but also the reason why the mail piece was incorrectly rated if such is the case. This serves as an excellent detection and thus deterrent mechanism because if a mailer or group of mailers consistently misrates the mail (for example, consistently utilizing an improper weight or use an incorrect rate table/calculation algorithm) this can be detected. The number and nature of the detected failures for the mailer or group of mailers to properly rate the mail pieces may be stored. The postal service can take appropriate action based on specific data as to the extent and reason for misrating by a mailer or group of mailers.

The present system can be made part of the meter recharging process wherein additional funds are entered into a metering system. This is to enable the continued printing of postage when the funds within the postage evidencing device 114 are exhausted. The verification that a current rate table or rate tables is installed in the metering system can be a requirement to enable recharging of the postage evidencing device 114 with additional funds. The postage evidencing device thus can only print a limited amount of postage or other value based on improper rating or obsolete rating tables. This amount is the amount of funds within a postage evidencing device between recharging operations. This limits the risk of a postal service due to rating with improper rate tables to the amount of funds currently in the meter system.

The downloading of current rate tables when made a part of the recharging operation and can employ downloading of the current rate table hash value. The hash value would be part of other information unique to the funds recharging transaction (or other funds transaction as for example,for current account meters, the

reporting of funds printed by the postage evidencing device since last audit) and may be encrypted to prevent tampering. The postage evidencing device 114 would reverify the rate table using the new hash value as part of the funds reset process. If the new hash value does not match the hash value computed from the resident rate table, no postage printing would be allowed. In an alternate arrangement, the postage evidencing device 114 would calculate the hash value in the current rate table and upload the device current rate table hash value to the data center before any funds recharging or other funds transaction is authorized. If the hash value from the postage evidencing device does not match the hash value calculated at the data center, no additional funds recharging (or the funds transaction) would be authorized by the data center. In either arrangement, the postage evidencing device 114 can display a message to the user indicating that updating the rate table is required.

It should be recognized that, rather than requiring the updating of the rate table or reverification of the rate table to be part of a recharging or other funds transaction, the requirement can be based on a calendar clock resident in the postage evidencing device 114. Thus, after a predetermined period of time, as for example twenty four hours, forty eight hours, seventy two hours or any other selected time period, the meter can become inoperative until a reverification that current rate tables are being utilized. In yet another arrangement this reverification can be at a point where particular value of postage has been printed or after a certain number of power up, power down cycles.

By requiring the uploading or recomputation of rate tables it is also possible to determine whether the rate table resident within the postage evidencing device has been tampered with because of the lack of appropriate hash value for either a current rate table or a previously valid rate table. In such case, meter operation can be inhibited either by the failure to enable recharging of the meter or by downloading a data code which inhibits operation of the meter.

It should still be understood that the arrangement described above in connection with insuring the integrity of the data loaded into the postage evidencing device 114 can be mailing data other information within the postage evidencing device 114 or peripherals to the postage evidencing device. For example, if a mailing list is downloaded into the postage evidencing device by the techniques described above, the hash values can be computed during the operation to insure the data was not corrupted during the loading process or the utilization of the data during operation of the postage evidencing device. The hash values can be generated each time a specified number of transactions (of any type) occur. The hash values would be stored in the postage evidencing or in the data center or other data repository. A postal service or a carrier or other party would thereby be able to detect and determine corruption of the data by querying the postage evidencing device or peripheral. The sequence of hash values stored would allow a determination of when and where tampering occurred depending on the nature of the parameters used to generate the hash value.

While the present invention has been disclosed and described with reference to the specific embodiments described herein, it will be apparent that variations and modifications may be made therein.

APPENDIX A

```
RT = 110290 120267 130248 140230 150242 160239 170233

1. Convert RT into binary value RTB

RTB = 0001 0100 0100 0001 1101 0001 0000 1000 0100 1110 1101 1100 1101 0101
0010 1110 1001 1011 1000 0010 1100 0100 1000 1100 0010 0101 0011 1000 1010
1010 1111 1101 0010 1011 1001

2. Take first 64 bits of RTB (B1) and encrypt it with DES key K=1234577777.
   Result is A1.

Leftmost 32 bits of output  = 3435858444 (CCCB0A0CH)
Rightmost 32 bits of output = 4259691368 (FDE5BB68H)

A1    =     CCCB 0A0C FDE5 BB68 (hex)

3. Take next 64 bits of RTB (B2) and calculate A1 XOR B2

A1    =     CCCB 0A0C FDE5 BB68 (hex)
B2    =     1D10 84ED CD52 E9B8 (hex)

A1 XOR B2 = D1DB 8EE1 30B7 52D0 (hex)

(A1 XOR B2)2      =     3,520,827,105     (decimal, leftmost 32 bits)
(A1 XOR B2)1      =       817,320,656     (decimal, rightmost 32 bits)

4. Encrypt A1 XOR B2 with DES key K=1234577777.  Result is A2.

Leftmost 32 bits of output  = 3323549928 (C61958E8H)
Rightmost 32 bits of output = 705585280  (2A0E6080H)

A2    =     C619 58E8 2A0E 6080 (hex)

5. Take remaining bits of RTB (B3) and calculate A3 = A2 XOR B3.

A2    =     C619 58E8 2A0E 6080 (hex)
B3    =     0000 0000 0000 0144 (hex)

A3 = C619 58E8 2A0E 61C4

(A2 XOR B3)2      =     3,323,549,928 (decimal, leftmost 32 bits)
(A2 XOR B3)1      =       705,585,604 (decimal, rightmost 32 bits)

6. Encrypt A3 with DES key K=1234577777.  The result is:

Leftmost 32 bits of output  = 1589293923 (5ERAB363H)
Rightmost 32 bits of output = 2709860754 (A1853192H)

RESULT = 5ERA B363 A185 3192 (hex)

       = 6,825,965,425,726,402,962 (decimal)
```

Claims

1. A postal rating system comprising:

   a postal rating device having non-volatile storage means;

   means for transmitting a postal rate table to said postal rating device such that said postal rate table is stored in said rating device non-volatile memory;

   means for transmitting to said postal rating device a hash code such that said hash code is stored in said rating device non-volatile memory, said hash code based on information from said rating table;

   means in said postal rating device for generating a hash code based on information from said re-

ceived rate table stored in said rating device non-volatile memory; and
means for comparing the received hash code with the generated hash code.

2. A postal rating system as defined in claim 1 wherein said transmitted hash code is an encrypted hash code and including means in said rating device for decrypting the encrypted hash code and comparing the decrypted hash code with the generated hash code.

3. A postal system as defined in claim 2 wherein the received hash code and the generated hash code are each based upon the entire rate table.

4. A postal system as defined in claim 2 wherein said transmitted hash code and said transmitted rate table each includes data as to the time period when the rate table is valid.

5. A postage evidencing device comprising:
means for storing a postal rate table in a non-volatile memory;
means for storing a hash code based on information from the rate table in said non-volatile memory;
means for receiving a request for printing of postage value;
means for recomputing the hash code from said information from said rate table stored in said non-volatile memory;
means for comparing the recomputed hash code based with said hash code stored in said non-volatile memory; and
means for comparing said recomputed hash code and said stored hash code.

6. A postage evidencing device as defined in claim 5 further including:
means for printing at least one of said stored and said recomputed printing hash codes on a mail piece;
means for printing said mail piece rating parameters on said mail piece such that a verifying party can reconstruct the rating process and determine if rating inaccuracy occurred.

7. A postage evidencing device as defined in claim 6 further including means for encrypting said hash code such that said printing means is enabled to print an encrypted hash code on said mail piece.

8. A system for verifying the accuracy of postal rating, comprising:
means for scanning a mail piece to detect a hash code printed on a mail piece and rating parameters also printed on the mail piece;
means for recomputing the rating process to determine the rating accuracy; and,
means for determining the correctness of said rating for said scanned mail piece.

9. A system as defined in claim 8 further including means for storing a profile of a mailer based on information from said determining means to provide data concerning rating activities for a series of mail pieces.

10. A mail piece having imprinted thereon a postal rate based on a postal rate table, the improvement comprising imprinting on said mail piece a code based on information derived from the postal rate table and which provides an identification of the rate table.

11. A mail piece as defined in claim 10 wherein said code imprinted on said mail piece is a value derived from processing said rate table information with a function which precludes recreating said rate table information based solely on said imprinted value.

12. A mail piece as defined in claim 10 wherein said code is encrypted.

13. A mail piece as define in Claim 11 wherein said function is a hash function.

14. A mail piece as defined in claim 13 wherein said code is encrypted.

15. A mail piece as defined in claim 13 wherein said code imprinted on said mail piece is related to a hash value.

16. A mail piece as defined in claim 15 wherein said code is an encrypted hash value.

17. A mail piece as defined in Claim 15 wherein the hash value is imprinted in machine readable form.

18. A mail piece as defined in claim 17 wherein said hash value is imprinted in bar code format.

19. A mail piece as defined in claim 18 wherein said bar code format is a bar half bar code format.

20. A mail piece as defined in claim 19 wherein said value is an encrypted hash value.

21. A method for postal rating, comprising the steps of:
transmitting a postal rate table to a rating device;
transmitting to said rating device a code, said code based on information from said rating table;
generating a code based on information from the received rate table; and
comparing the received code with the generated code.

22. A method as defined in claim 21 wherein said received code and said generated code are hash code.

23. A method as defined in claim 22 wherein said transmitted hash code is an encrypted hash code and including the further steps of decrypting the encrypted hash code and comparing the decrypted hash code with the generated hash code.

24. A method as defined in claim 21 where the transmitted and said generated codes are based upon the entire rate table.

25. A method of printing postage evidence, comprising the steps of:
storing a postal rate table in a non-volatile memory;
storing a code based on information from the rate table in said non-volatile memory;
receiving a request for printing of postage value;
recomputing the code from said information from said rate table stored in said non-volatile memory; and
comparing said recomputed code and said stored code.

26. A method as defined in claim 25 wherein said stored code and said recomputed code are each hash codes.

27. A method of printing postage as defined in claim 25 further including the steps of:
printing said code on a mail piece;
printing said mail piece rating parameters on said mail piece to enable reconstruction of the rating process from information imprinted on said mail piece.

28. A method as defined in claim 27 wherein said code is encrypted and said encrypted code is printed.

29. A method as defined in claim 25 further including printing a postage rate, printing the date of printing the postage rate and printing said code on said mail piece, said code containing data as to the time period when said rate table is valid.

30. A method as defined in claim 29 wherein said code is encrypted and said encrypted code is printed.

31. A system for verifying the accuracy of postal rating, comprising the steps of:
scanning a mail piece to detect a code for a mail piece printed on said mail piece and rating parameters also printed on said mail piece;
recomputing the rating process to determine the rating accuracy; and
determining the correctness of said rating for said scanned mail piece.

32. A method as defined in claim 31 wherein said code is a hash code.

33. A method as defined in claim 32 where in said code is an encrypted code and including the further steps of decrypting said encrypted code.

34. A system as defined in claim 31 further including storing a profile of a mailer or group of mailers based on scanned data concerning rating activities for a series of mail pieces for said mailer or group of mailers.

35. A method as defined in claim 23 wherein said transmitted hash code and said transmitted rate table each

include data as to the rate table validity time period.

36. A postal rating system comprising:
a postal rating device having secure storage means;
means for transmitting a postal rate table to said postal rating device such that said postal rate table is stored in said rating device secure storage means;
means for transmitting to said postal rating device a hash code such that said hash code is stored in said rating device secure storage means, said hash code based on information from said rating table;
means in said postal rating device for generating a hash code based on information from said received rate table stored in said rating device secure storage means memory; and
means for comparing the received hash code with the generated hash code.

37. A postal rating system as defined in claim 36 wherein said transmitted hash code is an encrypted hash code and including means in said rating device for decrypting the encrypted hash code and comparing the decrypted hash code with the generated hash code.

38. A postal system as defined in claim 37 wherein the received hash code and the generated hash code are each based upon the entire rate table.

39. A postal system as defined in claim 37 wherein said transmitted hash code and said transmitted rate table each includes data as to the time period when the rate table is valid.

40. A method of printing postage evidence, comprising the steps of:
storing a postal rate table;
storing a code based on information from the rate table;
receiving a request for printing of postage value;
recomputing the code from said information from said stored rate table; and
comparing said recomputed code and said stored code.

41. A method as defined in claim 40 wherein said stored code and said recomputed code are each hash codes.

42. A method of printing postage as defined in claim 40 further including the steps of:
printing said code on a mail piece; and,
printing said mail piece rating parameters on said mail piece to enable reconstruction of the rating process from information imprinted on said mail piece.

43. A method as defined in claim 42 wherein said code is encrypted and said encrypted code is printed.

44. A method as defined in claim 40 further including printing a postage rate, printing the date of printing the postage rate and printing said code on said mail piece, said code containing data as to the time period when said rate table is valid.

45. A method as defined in claim 44 wherein said code is encrypted and said encrypted code is printed.

46. A method for a mailing system, comprising the steps of:
generating a request for recharging a postage evidencing device with additional postage value to be printed;
determining the validity of a rate table associated with said postage evidencing device; and,
enabling recharging of said postage evidencing device if said rate table is determined to be valid.

47. A method as defined in claim 46 wherein said steps of determining includes said postage evidencing device transmitting to a remote location a hash code value of a rate table currently associated with said postage evidencing device.

48. A method as defined in claim 46 wherein said steps of determining includes transmitting to said postage evidencing device a hash code value of a currently valid rate table.

49. A method for a mailing system, comprising the steps of:
determining the validity of a rate table associated with a postage evidencing device; and
enabling operation of said postage evidencing device if said rate table is determined to be valid.

50. A method as defined in claim 49 wherein said determining step is initiated periodically based on a calendar clock value in said postage evidencing device.

51. A method as defined in claim 49 wherein said determining step is initiated based on the amount of postage printed by said postage evidencing device.

52. A method for a mailing system, comprising the steps of:
determining the validity of mailing data associated with a postage evidencing device; and
enabling operation of said postage evidencing device if said mailing data is determined to be valid.

53. A method as defined in claim 52 wherein said determining steps includes generating a hash code value based on said mailing data.

54. A method as defined in claim 53 wherein said hash code value is respectively generated and stored for later retrieval and verification.

55. A method as defined in claim 54 wherein said hash code values are stored in a secure memory.

FIG. 1

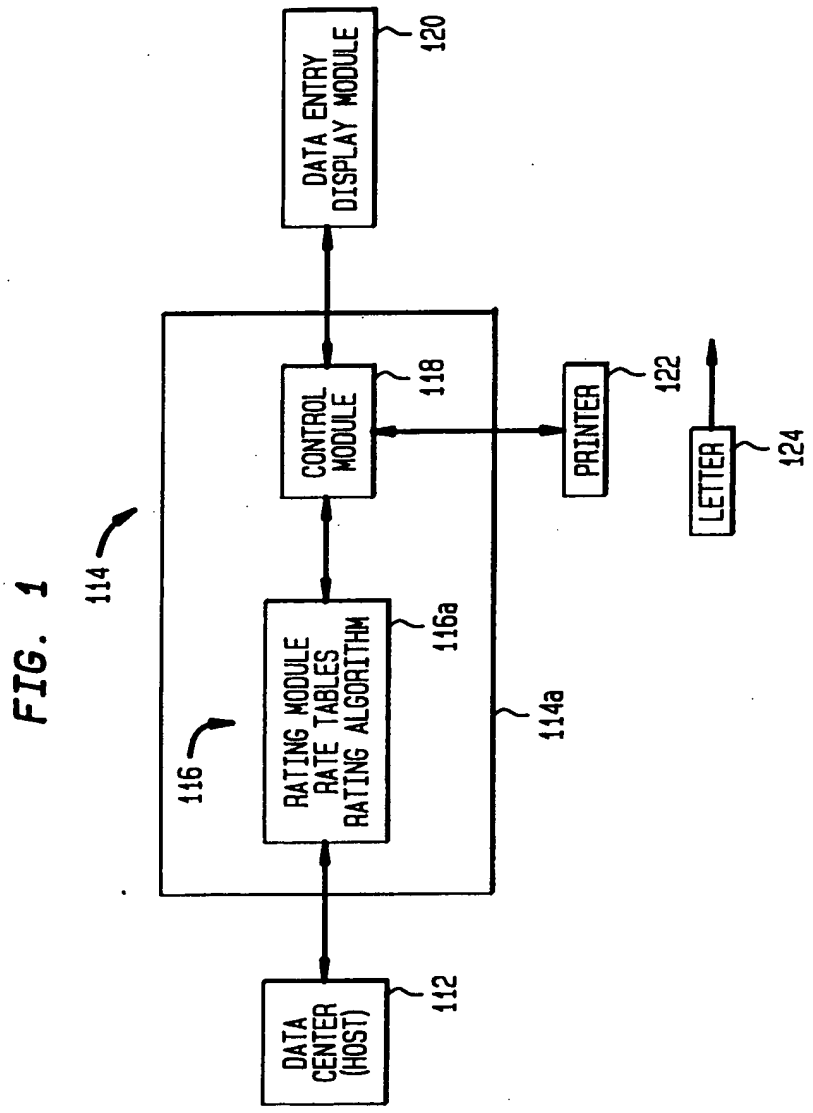

DATA ENTRY DISPLAY MODULE
120
CONTROL MODULE
118
PRINTER
122
LETTER
124
114
RATING MODULE
RATE TABLES
RATING ALGORITHM
116a
116
114a
DATA CENTER (HOST)
112

FIG. 2
DATA CENTER ACTIVITIES

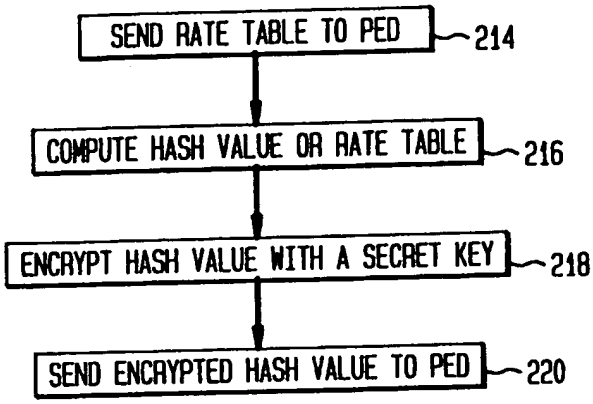
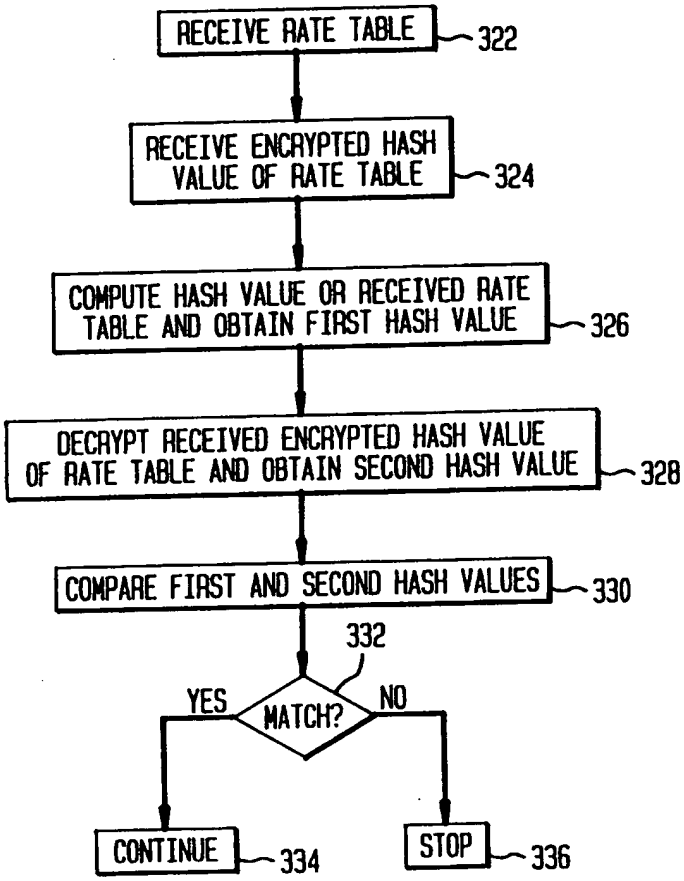


FIG. 3
POSTAGE EVIDENCING DEVICE ACTIVITIES



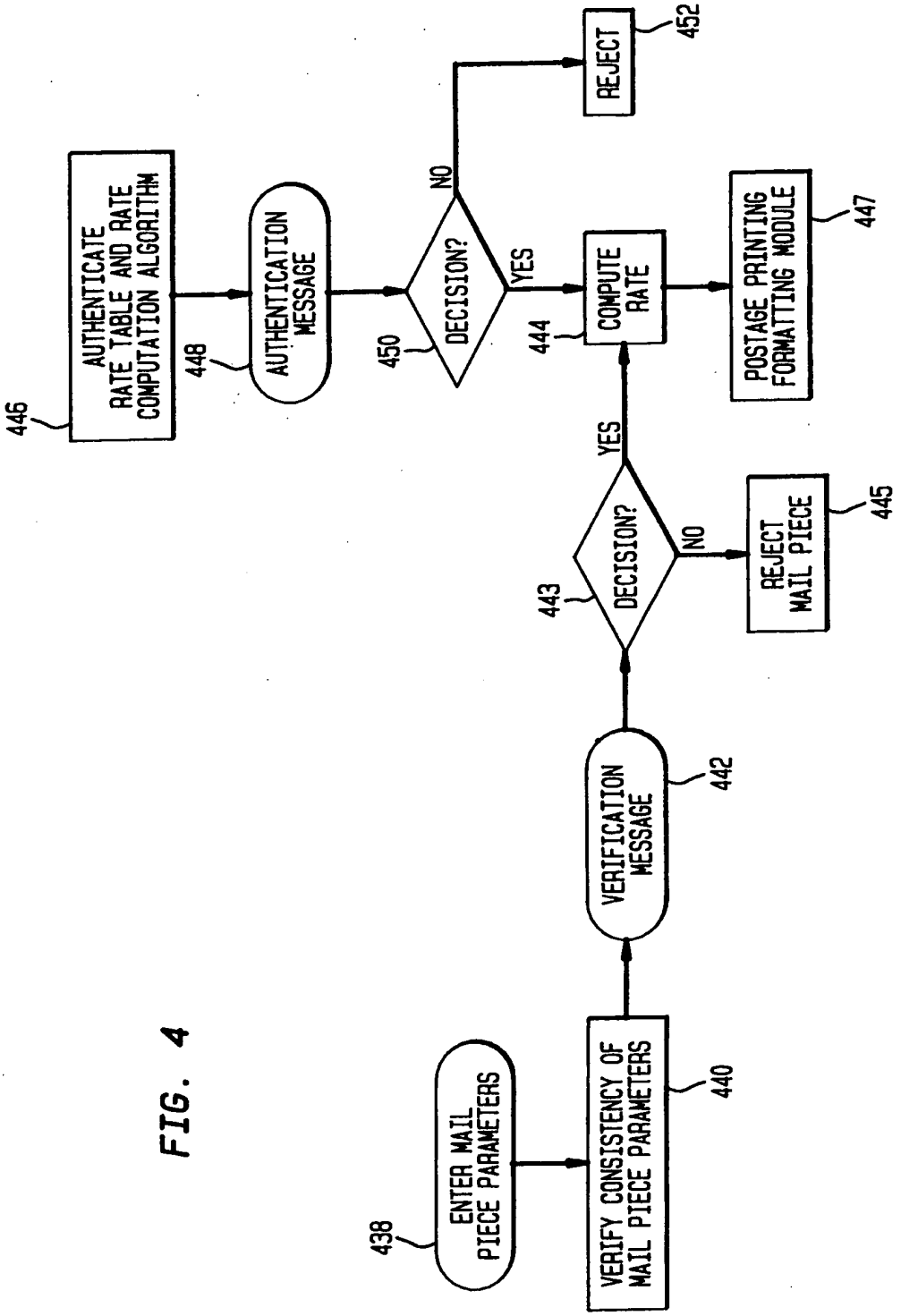

ENTER MAIL PIECE PARAMETERS
438
VERIFY CONSISTENCY OF MAIL PIECE PARAMETERS
440
VERIFICATION MESSAGE
442
DECISION?
443
NO
REJECT MAIL PIECE
445
YES
COMPUTE RATE
444
POSTAGE PRINTING FORMATTING MODULE
447
DECISION?
450
NO
REJECT
452
YES
AUTHENTICATION MESSAGE
448
AUTHENTICATE RATE TABLE AND RATE COMPUTATION ALGORITHM
446

FIG. 4

FIG. 5

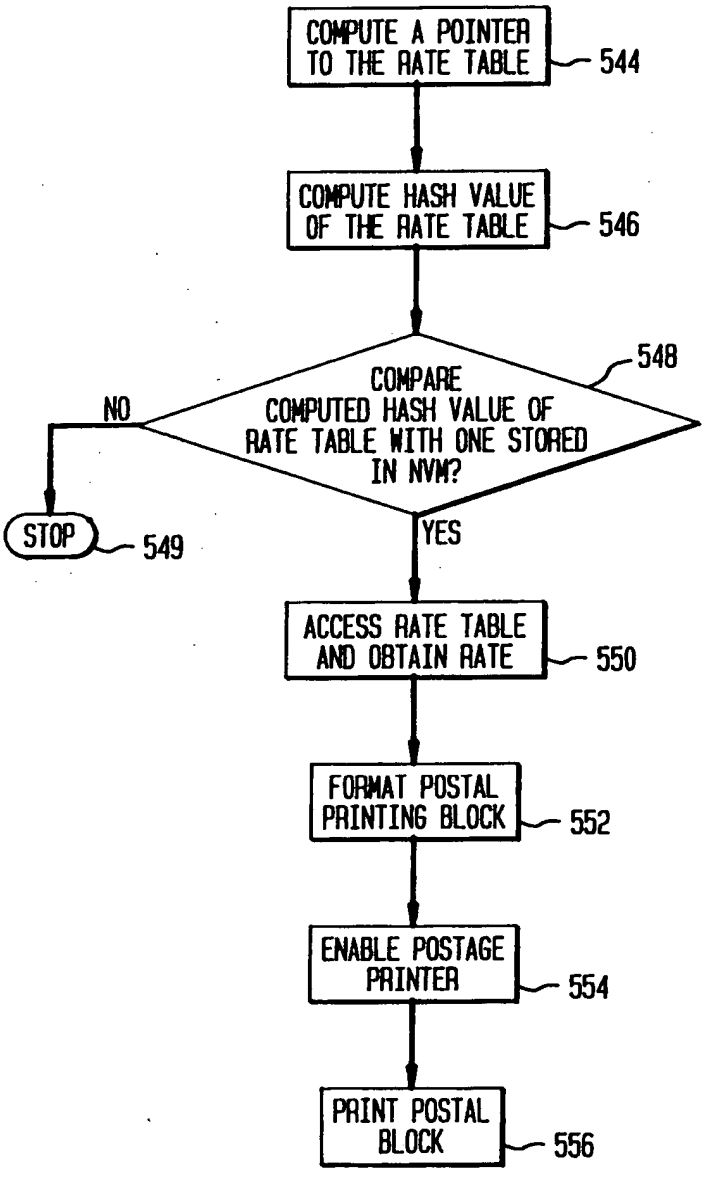
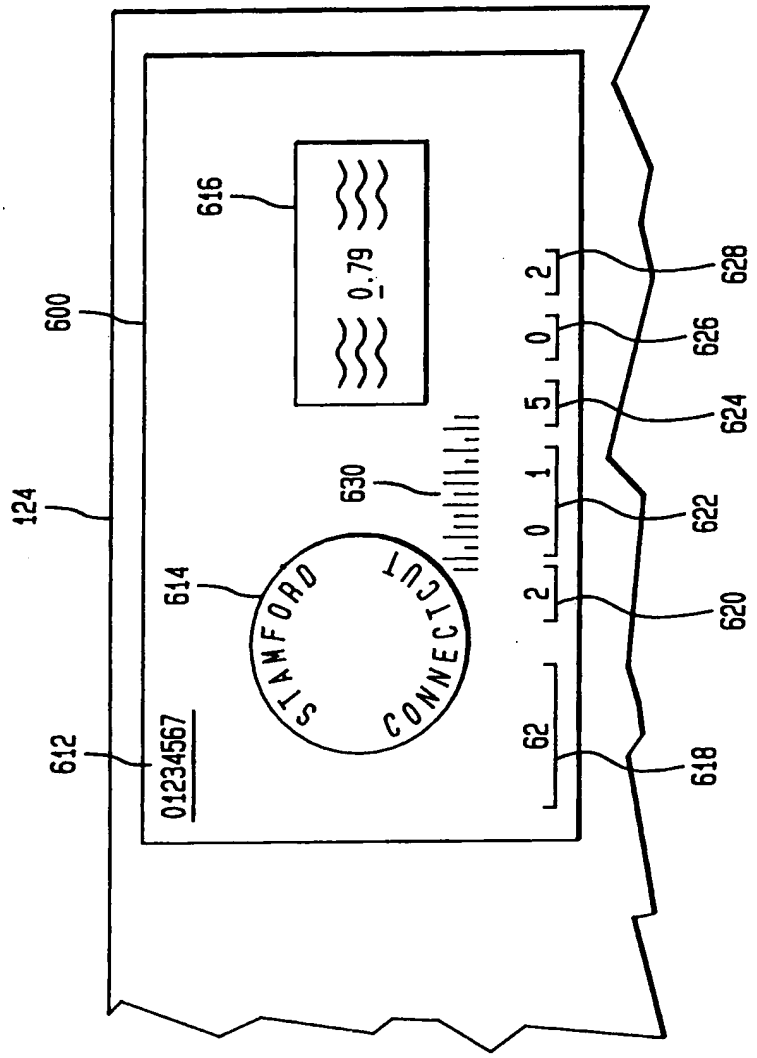

COMPUTE A POINTER TO THE RATE TABLE
544
COMPUTE HASH VALUE OF THE RATE TABLE
546
548
COMPARE COMPUTED HASH VALUE OF RATE TABLE WITH ONE STORED IN NVM?
NO
YES
STOP
549
ACCESS RATE TABLE AND OBTAIN RATE
550
FORMAT POSTAL PRINTING BLOCK
552
ENABLE POSTAGE PRINTER
554
PRINT POSTAL BLOCK
556

FIG. 6


600
616
0.79
628
626
624
622
620
618
630
124
614
STAMFORD CONNECT CUT
612
01234567
62
2
0
5
1
0
2

```
Set     Items   Description
S1     336962   VERIFY??? OR VERIFI?????? OR CHECK??? OR VALIDAT???
S2    1181198   COMPAR???? OR MATCH??? OR EQUAL??? OR UNEQUAL??? OR SYNCHR-
            ON???? OR UNSYNCHRON???? OR ASYNCHRON????
S3      14853   (DATABASE OR REFERENTIAL OR ENTITY OR KEY OR DOMAIN OR COL-
            UMN OR TABLE)(3N)INTEGRITY OR DATABASE(3N)(CHANGE OR CHANGING
            OR CHANGED OR UPDAT??? OR ALTER??? OR MODIFY OR MODIFI?????? -
            OR EDIT???)
S4    1029654   HASH??? OR MD5 OR CRC32 OR COMPRESS??? OR GZIP OR BZIP2 OR
            WINZIP OR ZIP OR REDUCED OR COMPACT OR MINIATURE OR ABRIDGED -
            OR SUMMARIZ???
S5    1508823   METADATA? ? OR META()DATA OR ATTRIBUTE? ? OR FIELD? ? OR P-
            ROPERT??? OR DATAFOUNDRY OR UML OR SQL92 OR DATA(2N)TYPE? ? OR
             SIZE OR LENGTH OR FILENAME? ? OR (FILE OR RECORD)()NAME? ? OR
             PARAMETER? ?
S6          5   S1(3N)S3(3N)(S1 OR S2)(3N)S4(3N)S5
S7         85   (S1(20N)S3(20N)(S1 OR S2)(20N)S4(20N)S5) NOT S6
S9         48   S7 NOT PD=(20020707:20060126)
S10   1918519   SECOND??? OR OTHER OR ANOTHER OR CURRENT OR NEW OR PRESENT
S11   2150176   PREVIOUS?? OR EARLIER OR PRIOR OR ORIGINAL?? OR EXISTING OR
             OLD OR PAST OR BEFORE???? OR PREEXISTING OR OLDER OR FIRST OR
             REFERENCE
S12       746   (S1(50N)S3(50N)(S1 OR S2)(50N)S10(50N)S4(50N)S5(50N)S11(50-
            N)S4(50N)S5) NOT (S6 OR S9)
S13       149   S12 AND IC=(G06F-007/00 OR G06F-017/00 OR G06F-012/00 OR G-
            06F-017/30)
S14       238   (S1(50N)S3(50N)(S1 OR S2)(50N)S4(50N)S5) NOT (S6 OR S9 OR -
            S13)
S15       112   S14 NOT PD=(20020707:20060126)
? show files
File 348:EUROPEAN PATENTS 1978-2005/Dec W04
         (c) 2006 European Patent Office
File 349:PCT FULLTEXT 1979-2005/UB=20051229,UT=20051222
         (c) 2005 WIPO/Univentio
?
```

```
? ds

Set     Items   Description
S1          0   AU=(VANOPDORP D? OR VANOPDORP, D? OR VAN()OPDORP D? OR VAN-
              ()OPDORP, D?)
? show files
File 347:JAPIO Nov 1976-2005/Aug(Updated 051205)
         (c) 2005 JPO & JAPIO
File 348:EUROPEAN PATENTS 1978-2005/Dec W04
         (c) 2006 European Patent Office
File 349:PCT FULLTEXT 1979-2005/UB=20051229,UT=20051222
         (c) 2005 WIPO/Univentio
File 350:Derwent WPIX 1963-2006/UD,UM &UP=200606
         (c) 2006  Thomson Derwent
```

```
Set     Items   Description
S1          0   AU=(VANOPDORP D? OR VANOPDORP, D? OR VAN()OPDORP D? OR VAN-
              ()OPDORP, D?)
? show files
File   2:INSPEC 1898-2006/Jan W1
         (c) 2006 Institution of Electrical Engineers
File   6:NTIS 1964-2006/Jan W3
         (c) 2006 NTIS, Intl Cpyrght All Rights Res
File   8:Ei Compendex(R) 1970-2006/Jan W3
         (c) 2006 Elsevier Eng.  Info. Inc.
File  34:SciSearch(R) Cited Ref Sci 1990-2006/Jan W3
         (c) 2006 Inst for Sci Info
File  65:Inside Conferences 1993-2006/Jan W4
         (c) 2006 BLDSC all rts. reserv.
File  94:JICST-EPlus 1985-2006/Nov W2
         (c)2006 Japan Science and Tech Corp(JST)
File  99:Wilson Appl. Sci & Tech Abs 1983-2005/Dec
         (c) 2006 The HW Wilson Co.
File 148:Gale Group Trade & Industry DB 1976-2006/Jan 26
         (c)2006 The Gale Group
File 636:Gale Group Newsletter DB(TM) 1987-2006/Jan 26
         (c) 2006 The Gale Group
```

```
Set     Items   Description
S1    3316967   VERIFY??? OR VERIFI?????? OR CHECK??? OR VALIDAT???
S2    9198901   COMPAR???? OR MATCH??? OR EQUAL??? OR UNEQUAL??? OR SYNCHR-
            ON???? OR UNSYNCHRON???? OR ASYNCHRON????
S3      68813   (DATABASE OR REFERENTIAL OR ENTITY OR KEY OR DOMAIN OR COL-
            UMN OR TABLE)(3N)INTEGRITY OR DATABASE(3N)(CHANGE OR CHANGING
            OR CHANGED OR UPDAT??? OR ALTER??? OR MODIFY OR MODIFI?????? -
            OR EDIT??? OR VIOLAT??? OR CORRUPT??? OR COMPROMIS???)
S4    4062224   HASH??? OR MD5 OR CRC32 OR COMPRESS??? OR GZIP OR BZIP2 OR
            WINZIP OR ZIP OR REDUCED OR COMPACT OR MINIATURE OR ABRIDGED -
            OR SUMMARIZ???
S5   12654397   METADATA? ? OR META()DATA OR ATTRIBUTE? ? OR FIELD? ? OR P-
            ROPERT??? OR DATAFOUNDRY OR UML OR SQL92 OR DATA(2N)TYPE? ? OR
             SIZE OR LENGTH OR FILENAME? ? OR (FILE OR RECORD)()NAME? ? OR
             PARAMETER? ?
S6          1   S1(3N)S3(3N)(S1 OR S2)(3N)S4(3N)S5
S7         96   (S1(20N)S3(20N)(S1 OR S2)(20N)S4(20N)S5) NOT S6
S8         87   S7 AND (PY<2003 OR PD<20020707)
S9   29692679   PREVIOUS?? OR EARLIER OR PRIOR OR ORIGINAL?? OR EXISTING OR
             OLD OR PAST OR BEFORE???? OR PREEXISTING OR OLDER OR FIRST OR
             REFERENCE
S10      1015   (S1(50N)S3(50N)(S1 OR S2)(50N)(SECOND??? OR OTHER OR ANOTH-
            ER OR CURRENT OR NEW OR PRESENT)(50N)S4(50N)S5(50N)S9(50N)S4(-
            50N)S5) NOT (S6 OR S8)
S11        26   S10 AND (S2 OR S3 OR S5)/TI
? show files
File 275:Gale Group Computer DB(TM) 1983-2006/Jan 19
         (c) 2006 The Gale Group
File  47:Gale Group Magazine DB(TM) 1959-2006/Jan 27
         (c) 2006 The Gale group
File  16:Gale Group PROMT(R) 1990-2006/Jan 27
         (c) 2006 The Gale Group
File 624:McGraw-Hill Publications 1985-2006/Jan 27
         (c) 2006 McGraw-Hill Co. Inc
File 484:Periodical Abs Plustext 1986-2006/Jan W4
         (c) 2006 ProQuest
File 613:PR Newswire 1999-2006/Jan 27
         (c) 2006 PR Newswire Association Inc
File 813:PR Newswire 1987-1999/Apr 30
         (c) 1999 PR Newswire Association Inc
File 239:Mathsci 1940-2005/Feb
         (c) 2005 American Mathematical Society
File 370:Science 1996-1999/Jul W3
         (c) 1999 AAAS
File 696:DIALOG Telecom. Newsletters 1995-2006/Jan 26
         (c) 2006 Dialog
File 621:Gale Group New Prod.Annou.(R) 1985-2006/Jan 27
         (c) 2006 The Gale Group
File 674:Computer News Fulltext 1989-2005/Oct W2
         (c) 2005 IDG Communications
File  88:Gale Group Business A.R.T.S. 1976-2006/Jan 24
         (c) 2006 The Gale Group
File 369:New Scientist 1994-2006/Aug W4
         (c) 2006 Reed Business Information Ltd.
File 160:Gale Group PROMT(R) 1972-1989
         (c) 1999 The Gale Group
File 635:Business Dateline(R) 1985-2006/Jan 26
         (c) 2006 ProQuest Info&Learning
File  15:ABI/Inform(R) 1971-2006/Jan 26
         (c) 2006 ProQuest Info&Learning
File   9:Business & Industry(R) Jul/1994-2006/Jan 26
```

037279

RDBMS server choice gets tougher

Relational DBMS Servers Buyers Guide

Comparing **products is difficult when feature sets are similar and servers must** match **application architecture.**

Byline: Michael Goulde and Wayne Eckerson
Journal: Network World Page Number: 52
Publication Date: May 23, 1994
Word Count: 4843 Line Count: 461
Section Heading: Opinions

Caption(s): Graphic, Reader views on relational DBMS servers, Susan Slater

Text:

With vendors continuously matching one another on a feature-by-feature basis, it is increasingly difficult to tell one relational database management system server package apart from another.

While product parity reduces the chance of making a bad purchasing decision, the selection process is anything but a cakewalk. Just when users are ready to go with one vendor that has certain advanced features, competitors roll out the same enhancement, which complicates the final decision.

It goes on all the time. For example, when Sybase, Inc. introduced stored procedures in the first release of SQL Server, it didn't take long for Oracle Corp. to put that feature into Oracle7, which was available after SQL Server. While Borland International, Inc. was first out with event alerters in its InterBase product, the feature didn't catch on until The ASK Group, Inc. added it in what is now the Ask-OpenIngres/Intelligent Database. And soon most vendors incorporated event alerters. Informix Software, Inc. introduced support for Binary Large Objects (BLOB) or unstructured data and soon a flood of vendors began supporting BLOBs.

And so it goes, even with performance claims. As soon as one vendor heralds a benchmark test showing that its product performs best, its competitors offer different benchmark results that put them at the top of the best performer list.

The advent of client/server computing makes the choice of a strategic relational DBMS server even trickier because users must decide how they will take advantage of the ability to break processing chores apart and run them on clients and servers.

Users should weigh how different relational DBMS servers permit them to split important functions between the client and server. This type of comparison is more important than the features a given product may provide. It is also important to examine the type of tools relational DBMSs offer for optimizing database processing as well as how easily data can be distributed across multiple servers.

Users that focus on client application functionality may find that the selection of a relational DBMS server is already made for them. Some vendors are bundling client applications and relational DBMS servers together, which is forcing some users to support multiple relational DBMS servers.

Zale Corp. is such a company. The firm enables departments to select applications that dictate the choice of a relational DBMS server even though it has chosen Sybase as its strategic vendor. ``We want to be applications-driven, not relational DBMS-driven,'' says Tom Carson, Zale's director of distributed systems in Irving, Texas. ``So we will implement any relational DBMS required to support an application chosen by our

users.''

For example, Zale's accounting group last year replaced a custom mainframe general ledger application with a prepackaged application from Oracle that required use of Oracle7. Yet Zale continues to standardize on Sybase's SQL Server whenever an application supports it because Zale's developers fell in love with that product's ease of use, Carson says.

There are also more relational DBMS servers hitting the market. For example, Microsoft Corp. recently announced a Windows NT version of its Microsoft SQL Server. With this new offering, Microsoft is positioning Windows NT as a direct competitor to Unix as the operating system of choice for relational DBMS servers, a strategy that will effectively force users to make an operating system and relational DBMS server decision concurrently.

In addition, Microsoft is essentially parting ways with Sybase. Under a strategic partnership, Sybase licensed the heart of its SQL Server to Microsoft, which then developed its own version of that product. The two companies are now going in different directions with their future SQL Server products. This just adds more confusion to an already confusing picture.

IBM has also made recent strategic moves in the relational DBMS server market. The firm rolled out DB2/6000, a version of its popular host-based DB2 that runs under Unix on IBM's RISC System/6000 platform. IBM announced it will port DB2/6000 to other Unix-based platforms and possibly even to Windows NT, a dramatic departure from its past strategy of sticking to IBM platforms only. Users who have held IBM databases as the data center standard now have the option of retaining IBM technology while implementing new architectures.

And other vendors just keep rolling out new versions and features. Oracle, for example, introduced a version of Oracle7 that will run on massively parallel systems, as well as a multimedia database tool that will support interactive video.

MARKET LOW DOWn

Clearly, there is a lot to choose from on the market today. Users downsizing from mainframe environments don't want to have to make compromises in selecting a relational DBMS server.

``(We want) the same degree of robustness we've become accustomed to in our mainframe databases,'' says Jonathan Markow, project leader at Columbia University's Administrative Information Services. ``All of the same fail-safe features, like commit, rollback, checkpoint, restart and restore, have to be available for any serious downsizing development.''

A lot of those capabilities are found in high-end products, such as Computer Associates International, Inc.'s family of CA-IDMS and CA-Datacom products, IBM's MVS-based DB2, Oracle's Oracle7 7.0, Software AG's Adabas 5.3.2 and Sybase's SQL Server 10. These products support very large databases, typically in the range of a hundred gigabytes to over a terabyte. They also support thousands of users and meet the most demanding performance requirements.

High-end products are typically designed to make optimum use of system resources that enable them to provide maximum scalability. They also run on multiprocessor architectures, address large amounts of memory that can exceed 512M bytes and support databases that span many physical disks.

Products in this segment have a broad feature set, including support for two-phase commit, data replication, stored procedures, triggers, on-line backup and typically lead the industry in the introduction of new features. They support high throughput on-line transaction processing (OLTP) applications with the utmost in data integrity and are often used for banking, financial, order processing, inventory and distribution system applications within Fortune 500 companies.

In the mid-range are such products as The ASK Group's AskOpenIngres/Intelligent Database, Borland's InterBase 3.3, Hewlett-Packard Co.'s HP ALLbase/SQL 6.0, IBM's RS/6000-based version of

DB2, Informix's Informix-OnLine 7.0, Microsoft's Micro-
soft SQL Server 4.2X and Progress Software Corp.'s Progress 7.

These products support databases that range up to a few hundred gigabytes and several hundred users. Their architectures can support multiprocessing and are used in smaller companies or de-partments of larger firms. They may be used for OLTP, but the transaction rates are likely to be lower than their high-end counterparts, and they scale to the highest end processors.

At the low end are such products as Btrieve Technologies, Inc.'s NetWare SQl 3.0, Gupta Corp.'s Gupta SQLBase Server 5.1.4 and Watcom International Corp.'s Watcom SQL Network Server 3.2. These products are typically used for workgroups or small departments. They usually run on single-processor systems, support less than 100 users and databases that are less than 1G byte in size.

There are three key factors to consider when evaluating relational DBMS server products: the client/server architecture they will be deployed in, how important features are implemented and the level of distributed DBMS support provided.

FITTING CLIENT/SERVER MODEL

The term client/server is so overused today that it has ceased to have a precise meaning. Client/server is simply a model for building information systems (IS) that are organized into discrete components deployed at different locations on local- and wide-area networks.

There are two dimensions to client/server architectures. One defines the clients, servers and networks used to build client/server applications. The other defines the internal architectures of the elements used in the client/server environment the internal workings of a relational DBMS server, for example.

Each dimension plays a role in selecting a relational DBMS server, although the application architecture will likely have already been defined before the purchase decision is made.

The original concept of client/server embraced a two-tiered model with desktop clients usually personal computers on one tier and servers often Unix relational DbMS servers on the other. To build applications in this two-tiered model, users have to decide whether presentation, application logic and data management functions should reside on the client or server.

The decision is easy for two of the three. Presentation, such as a graphical user interface (GUI), is the province of the client, while data management is the server's domain. Making the third decision where to put the application logic poses a problem.

If application logic is placed on the client, the server is freed from processing that task; thus, the application should perform faster. But then the problem arises of how to ensure that business rules defining a company's policies and procedures for completing business tasks are enforced and that client applications are up-to-date. On the other hand, placing application logic on the server can be a step back to traditional monolithic applications that are difficult to change and extend.

Current thinking is that a three-tier model, similar to the ANSI 3 schema database model, is a more flexible way to approach client/server architectures. This model has a physical storage and management layer and an external presentation layer. Between these layers lies a representation of the physical data based on the logical model of information that users require to do their jobs. This middle layer translates the complexities of the relational model to concepts that end users can understand.

The middle layer has application program interfaces (aPI) that translate between what the user wants to get done and how it gets done in the physical database. Business rules might be enforced in the middle layer or at the back-end server, but this decision is very application-specific.

The advantage of a three-tier approach is that the implementation of the physical layer may change without affecting the developer's or user's view of the data. This is not always the case when the traditional two-tier

client/server model is used because the client application is more likely to use database-specific syntax, semantics and features that are not transferable to **other** products.

Aside from choosing a client/server model, there are also issues of how a client application can access data on disparate databases. This can be done using gateways, APIs, such as Microsoft's Open Database Connectivity (ODBC), or even such tools as the SQL Access Group's Call-Level Interface, which provide some translation functions. These functions can be done on the client, the server or a combination of the two (see story, page 55).

Both these models can be implemented with any relational DBMS server product. The selection of a specific product hinges on how large the database is and how many users will be supported. It is possible to have a two-tier architecture that requires a high-end server if there are hundreds of users and a terabyte of data, or low-end server if there are hundreds of megabytes of data and only a dozen users.

Architectures for the internal workings of relational DBMS server products have been evolving over time (see story, page 58). Early relational DBMS servers, such as Informix's Informix-OnLine 5.01, Oracle's Oracle6, Borland's InterBase 3.3 and Sybase's SQL Server 4.8, required each client to communicate with dedicated server processes.

Vendors then developed a multiserver, multithreaded architecture that enables groups of clients to dump requests for database processing into a queue and have those requests distributed to the next available server process for completion. The multiserver, multithreaded architecture was first available in Ingres' Intelligent Database 6. 4, which was subsequently acquired by The ASK Group, and Progress' Progress 6. It is now used in Informix's Informix-OnLine 7.0, Oracle's Oracle7 with multithreading option and Sybase's Virtual Server Architecture.

AT THE CORE

Overall, most relational DBMS servers have similar high-level features, differing only in minute details of implementation. The absence of any key feature, however, is extremely important.

How well a relational DBMS server performs is one of its most important features, particularly as the **size** of the database increases and more users access it. Among the methods for providing top-notch performance are various indexing schemes and query optimization techniques.

Indexing methods include **hashed** indexes and clustered indexing. These methods are designed to provide faster access to data. For example, **hashed** indexes use mathematical transformations of an actual key data- **field** value to point to records in the database. Clustered indexes sort rows of data on an ongoing basis so their physical order on a disk is the same as their logical order in the index. Products that support more methods provide greater flexibility in creating indexes that will optimize database performance.

Query optimization is a somewhat esoteric technology. A query optimizer uses special algorithms to determine the optimum method for retrieving data, developing an execution plan and then carrying out that plan. The relational DBMS server should be able to compile and save the plans for executing frequently repeated queries.

A query optimizer is among the most important and proprietary of a relational DBMS server vendor's technologies and one that contributes significantly to performance. Methods of query optimization include cost-based, rules-based and table statistics.

Cost-based optimization looks at different methods of getting the requested data and evaluates such factors as the type of indexes that are available, how much data must be moved across the network, the CPU cycles required to complete the query and available CPU cycles. It will then choose the method with the lowest overall resource requirement. A cost-based optimizer that uses table statistics is the most sophisticated approach.

Rules-based optimization uses a more theoretical approach to optimizing

queries. Optimizers that use table statistics actually consider how many rows and columns are in a table, how many bytes are in each row and other statistics to develop a plan for completing the query.

Many relational DBMS servers have an ``Explain'' capability that shows the database administrator how a query will be executed. Explain may show a graphical picture of how different data tables will be joined and which keys will be used to join the tables. The database administrator can use this explanation to manually tune large, time-consuming or repetitive queries.

Sybase and Oracle support the most methods of optimization. Btrieve and Empress Software, Inc. only support optimization-based query syntax and rules.

Just as performance is an issue in determining how quickly client requests are processed, **referential integrity** features are important for keeping data that is shared across a set of related records updated. **Referential integrity** becomes more important as the number of different client applications capable of updating the data grows.

Referential integrity maintains consistency in related data stored in different tables. A record in one table that may contain customer information can have related records in an order table stored elsewhere. If the customer record is deleted, the order records also have to be deleted or they will be orphans. Likewise, if the customer information is changed, the order table must also be changed.

One method of maintaining referential integrity is to use triggers, actions that the database carries out automatically in response to an insert, update or delete event. Triggers may be executed on the client or server. In some cases, triggers can be executed across multiple nodes on the network.

The advantages of executing triggers on the server is that they are easier to maintain and they insure that all applications abide by their requirements. When executed on the client, performance improves but integrity is more difficult to maintain.

Client-side referential integrity is less desirable because there is no way to guarantee that all client applications are imposing the same trigger or are running the current trigger version. Between the two approaches, server-side triggers are preferable because of the control they offer to insure that referential integrity is being enforced.

Most vendors support server triggers, but Progress' Progress 7 provides the option of executing triggers on the client or server. About the only differences in how triggers are implemented are in the number of triggers supported and whether those triggers can execute before or after a record is updated. There are no standards for triggers, and they are not portable from one product to another.

In addition to triggers, referential integrity can be maintained through declarative statements, an ANSI SQL standard that calls for a means of enforcing referential integrity through primary and foreign keys and SQL statements. Oracle, IBM with its DB2/2, Gupta, Watcom and XDB Systems, Inc. are among the vendors with declarative statements.

Another important feature to examine is stored procedures, code that allows business rules and other application logic to be maintained in the database. Stored procedures may be written in third-generation languages, as is the case with IBM's DB2/6000, or fourth-generation languages, as is the case with Sybase and Oracle.

Stored procedures can perform many different operations and can be executed on the client or server. For example, a stored procedure in an order entry application can be used to **check** a customer's account balance maintained in another application before accepting an order. A stored procedure in a transaction processing application can **check** a credit limit maintained in an accounting application before committing a transaction.

Execution of stored procedures on the server is more common because it

allows for centralized maintenance and control. Any stored procedure is suitable for the server. However, this can often lead to applications that are almost as monolithic as traditionally designed ones.

Execution on the client allows flexibility but at the expense of possible inconsistencies when applying business rules. Client-side stored procedures are ones that are most directly related to an action a user may perform, such as entering or editing data.

Today, each product has different semantics and syntax for invoking stored procedures and different server implementations preventing portability of stored procedures across products. Efforts are under way to standardize syntax and semantics, such that a client's application code does not have to be modified to invoke a stored procedure on different servers. That effort may bear fruit next year.

Another important feature is event alerters that enable the database to execute code or a stored procedure in response to some change. They are more general than triggers, and the event might be based on a calculation, update or attempt to delete data.

The event alerter can provide real-time notification to an application that a particular event has occurred. This way, an application doesn't have to constantly poll the database to see if a parameter has changed, thus avoiding the performance burden that polling places on the network, server and client.

For example, an event alerter might go off if the inventory level of a particular product went below a certain level. A notification could be sent to an application that could automatically place an order for more of that product from the supplier.

Access to electronic mail APIs, such as the ones Microsoft has implemented in its SQL Server, expand event alerter notification options. For instance, an event alerter cannot only order more product, but also send an E-mail message that informs the person in charge of tracking inventory that the order was placed.

Having event alerters in a relational DBMS server adds to the flexibility a developer has in designing applications. However, like stored procedures, these are not portable today across databases from different vendors.

Borland was the first to implement event alerters. The ASK Group was next, followed by Digital Equipment Corp., Gupta, Oracle, Progress, Sybase and UniSQL, Inc.

In a multiuser environment, the need to maintain the integrity of data so that two users cannot change the same row or field simultaneously has to be balanced against performance requirements. Relational DBMS servers use different combinations of data-locking levels, lock types and data isolation to prevent changes being made concurrently; to keep two users from locking each other out of the database, in what is called a deadly embrace; and to keep to a minimum the time spent waiting for a record to be unlocked.

A database can be locked at different levels, ranging from a page level that is typically measured in 2,000 bytes a page to the field level. The finer grain the lock is, the less likely

users are to be prevented from updating particular data.

Most products lock at the page and table levels. Only Sybase, microsoft, Gupta, HP in its HP Image/SQL 6.0 and IBM in its MVS-based DB2 don't support row locks.

Field-level locking is difficult to implement, and only a few vendors (Borland, Software AG, Empress and Digital) implement it today.

Tools for managing relational DBMS servers are an area in which most products are weak today. Managing multiple distributed relational DBMS servers with hundreds of clients making requests and updates can be very difficult and often requires logging on to each individual server. While this can usually be done remotely, it means individually administering each server instead of as a group or domain of servers.

What users require is the ability to manage multiple servers in a consistent, integrated fashion, ideally with a GUI. While there are third-party products now appearing that provide this functionality, the Windows NT-based version of Microsoft's Microsoft SQL Server provides one of the most graphical and integrated ways to manage distributed servers.

DiSTRIBUTED DATABASES

One of the **new** battlegrounds for relational DBMS servers is the area of distributed database management. During the past two years, most vendors have added or announced support for distributed capabilities, such as distributed query processing, two-phase commit and data replication.

The key limitation in building a distributed database is the network. Trying to update or query data across a WAN will result in slow response times, especially if users are accessing large volumes of data in multiple remote sites. Also, network links can fail, preventing users from accessing remote data and potentially undermining the integrity of distributed updates.

There are two basic strategies for building and maintaining distributed databases: two-phase commit and data replication. Each addresses the limitations imposed by networks differently. Two-phase commit emphasizes data integrity at the expense of performance, while replication emphasizes reliability at the expense of timeliness. Two-phase commit coordinates transactions across multiple relational DBMS servers in real time. The protocol ensures that every database is updated. If one database is not ready to participate in the transaction because it is off-line or the network has crashed, two-phase commit rolls back the transaction in all the other databases and starts the process again.

However, two-phase commit imposes a significant performance penalty. The protocol creates a lot of network overhead since each of the databases participating in a transaction must constantly communicate with one another. Also, two-phase commit operations often result in aborted transactions, which forces users to try completing the transaction again, thus taking up even more network capacity.

Most vendors listed in the Buyer's Guide chart on page 57 now support two-phase commit, with the only major exceptions being Gupta and HP in its HP ALLbase/SQL 6.0. IBM supports two-phase commit only in its MVS-based and AS/400 versions of DB2.

Two-phase commit is helpful when distributed data used in financial and airline reservation system applications must be kept synchronized in real time at all costs.

Cincom Systems, Inc. recently introduced a new type of two-phase commit that provides better performance and higher availability of database resources. The latest version of its Supra Server supports a two-phase commit protocol called Transaction Partners that lets applications complete a transaction even if one or more databases are unavailable. The Transaction Partners protocol uses an optimistic roll-forward mechanism to coordinate updates among multiple databases, whereas most two-phase commit protocols employ a pessimistic, roll-back method.

In the Transaction Partners protocol, each database participating in an update stores information about the update in a log. If a database fails in the middle of an update, that database, upon recovery, will check other databases that participated in the transaction to see if any completed the update. If it finds one database that successfully executed the transaction, it then completes the update.

To get around some of the performance problems associated with two-phase commit, vendors have embedded replication facilities in their products or offer replication add-on modules.

Unlike two-phase commit, replication isn't very vulnerable to network or machine outages. With replication, copies of updates are distributed to one or more remote databases. Updates can be replicated from a master database to multiple remote databases or vice versa. Users can also specify the time intervals for replication to occur, such as every 30 seconds or

once a day.

Users are deploying replication to create backup copies of their databases and consolidate reporting information from branch offices to a central site. Some are also using replication to create information warehouses for decision support purposes as well as to coordinate updates between legacy and downsized applications.

In reality, replication is not a new capability. Users have written replication utilities for many years to solve numerous in-house requirements. However, users are more than happy to turn such development work over to vendors that can offer such utilities less expensively and provide support and maintenance, as well.

Last year, Software AG became one of the first vendors to ship replication capabilities. It has been followed by a host of others, including The ASK Group, Digital, HP, IBM, Informix, Oracle and Sybase.

However, users should realize that not all vendors have taken the same approach to replication. Sybase, for example, provides Replication Server as an add-on option to SQL Server 10. Also, the company's Replication Server supports asynchronous replication. That is, it distributes or replicates individual transactions from a primary database to one or more remote databases using a store-and-forward mechanism.

Replication Server captures transactions in a log at the central site and then delivers them when the remote servers are available. However, the recipient databases are read-only and can't be updated directly by local users. Users at the remote sites can update replicated data by executing a stored procedure against the primary database, which then replicates the changes back to the remote site.

Oracle's Oracle7 relational database, on the other hand, supports synchronous replication in which entire database tables, not individual transactions, are replicated. Also known as table-driven replication, this feature takes snapshots of the database, and it lets users replicate a table in a master database to a table in one or more remote databases. It also lets users replicate multiple tables in a master database to a single table in remote databases. Unlike users of Sybase's Replication Server, Oracle users can schedule the intervals at which replication will take place.

The service and support provided by a vendor may ultimately become one of its primary distinguishing characteristics. Hyatt Hotel Corp. chose to implement Informix's Informix-OnLine relational DBMS server several years ago because of its front-end tools and the support level Informix was willing to provide in helping Hyatt move all its applications from a mainframe to Pyramid Technology Corp. multiprocessing Unix machines.

Informix provided a half-dozen programmers to help Hyatt's staff rewrite several old COBOL and Assembler programs using the Informix fourth-generation language. Informix's staff also helped Hyatt tune the performance of the Informix relational DBMS server on the Pyramid multiprocessors.

``Establishing a strong partnership with Informix was a big reason for our downsizing success,'' says Gordon Kerr, director of IS at Hyatt in Oak Brook Terrace, Ill.

THE BOTTOM LINE

The most critical step in selecting a relational DBMS server is to thoroughly understand your application requirements first. Selecting the relational DBMS server first can seriously constrain development, restrict functionality and impact performance.

However, widespread support of such standards as ODBC that provide access to disparate databases as well as portable stored procedures will make it easier to swap out relational DBMS servers as the need arises. Nevertheless, client/server applications are difficult to build and deploy, and selecting an inappropriate server can sink a project before it even starts. Prototyping and benchmarking are critical steps to ensure overall success.

While today's products have similar feature sets, the application of relational DBMS servers in new areas will require additional features. For example, Rick Fortin, vice president of information services at Label Art, Inc. in Wilton, N.H., says a critical factor for relational DBMS servers is that they support nested or post-relational capabilities. That is, relational DBMS servers should provide support for multiple entries within a single field, such as two or three telephone numbers.

This runs contrary to relational DBMS server theory, which requires database administrators to normalize data as much as possible. However, nested relational databases are more practical because they represent the real world more accurately, Fortin says.

Looking further into the future, emerging object-oriented DBMSs (OODBMS) will challenge relational DBMS servers as the next wave of technology for certain applications using complex data types. Fitted with SQL interfaces, these technologies are maturing in products that will challenge relational DBMS servers in some areas. Already, OODBMS vendors are calling relational products the legacy systems of tomorrow.

Goulde is editor in chief of the Unix in the Office newsletter, and Eckerson is senior editor of the Open Information Systems newsletter at Patricia Seybold Group, Inc., a Boston-based research and consulting firm. They can be reached at (617) 742-5200.

02010288 SUPPLIER NUMBER: 18842592 (THIS IS THE FULL TEXT)
Oracle. (Oracle7 7.3 and Universal Server) (one of six database server evaluations in " Comparison **Summary") (DBMS Server** Comparison **Supplement) (Software Review) (Evaluation)**
Rennhackkamp, Martin
DBMS, v9, n12, pS12(3)
Nov, 1996
DOCUMENT TYPE: Evaluation ISSN: 1041-5173 LANGUAGE: English
RECORD TYPE: Fulltext; Abstract
WORD COUNT: 2298 LINE COUNT: 00190

ABSTRACT: Oracle's Universal Server, including Oracle7 7.3, has caught up with its competitors in technical quality by integrating relational database management with complete Web, messaging, and multimedia functionality. Numerous performance improvements and excellent Web support make Oracle's offerings among the market's best. The included Oracle WebServer facilitates the development of interactive applications that both reflect and impact their associated databases. Integrated support for images, text, multimedia, messaging, and analytical data makes Universal server very extendable. Strong parallel and parallel management systems, good security features, and a focus on network computing, as well as online transaction processing (OLTP) and online analytical processing (OLAP) make the Universal Server a comprehensive package.

TEXT:

ORACLE FINALLY PUTS ITS MONEY WHERE ITS MOUTH IS WITH ORACLE7 RELEASE 7.3 AND THE UNIVERSAL SERVER.

With a marketing hype rivaled only by Microsotf's Windows 95 launch, the Oracle Universal Server, including Oracle7 Release 7.3, was forced onto the Oracle user community and the general IT public at large at the Oracle Developers' Conference in February 1996. The aim of the Oracle Universal Server is to combine the functionality of the world's bestselling client/server relational DBMS with complete Web, text management, messaging, and multimedia information servers. The multimedia services include fully integrated relational, spatial, text, audio, and video information to any user with a standard Web browser accessing a Web-enabled database. The Oracle Universal Server is targeted to give Oracle users a comprehensive platform designed to handle all of these needs: network-centric computing, mission-critical online transaction processing (OLTP), online analytical processing (OLAP), and data warehousing applications. In this review I investigate what the Oracle Universal Server has to offer.

Relational Data Model

Like the **other** DBMSs that claim to be relational, Oracle also supports the relational data model to an extent. Data is stored in tables, with integrity constraints to protect the correctness of the data.

You define standard declarative referential, key, and column integrity constraints as part of the CREATE TABLE and ALTER TABLE statements, using an ANSI SQL-92-like syntax. For referential constraints, Oracle supports the RESTRICT (NO ACTION is the correct ANSI SQL-92 syntax) and CASCADE options for delete operations on the primary table, but it supports only the RESTRICT option for updates on the primary table. Oracle also supports the MATCH NONE, MATCH FULL, and MATCH PARTIAL options fOr composite and nullable foreign keys, which few other DBMSs do at this stage.

Alternatively, you can code your own integrity constraints using Oracle's powerful triggers. Oracle's pre-operation and pre-row triggers are

an efficient way to check whether an operation will violate a **referential - integrity** constraint before the operation is actually performed.

Database Objects

The Oracle Universal Server has integrated support for relational and multimedia types of data, including static images, video, audio, text, spatial, messaging, and analytical data. The Oracle Universal Server is very extendable. You can start with the basic Oracle7 Release 7.3 database engine and add the options you need for your particular environment:

* The Oracle ConText option lets Oracle handle large units of unstructured text the same way it handles the standard, structured data types. The ConText option provides SQL-like interface to access the textual data, which makes it possible to apply the ANSI SQL-92 textual data type functions to large volumes of unstructured text. This makes it easy to extend existing applications to perform intelligent text searches and text reductions. The ConText option also provides facilities for text retrieval, classification, and management.

* The Oracle Video Server lets you store, manage, and display rich multimedia data to clients over a network. This includes full-motion, full-screen video as well as high-fidelity audio. This option includes not only the Orate Video Server but also the Oracle Video Client and Oracle Media Net, which can deliver video streams at varied bit rates in a distributed environment.

* The Oracle Spatial Data Option manages spatial (geographic) information as an integrated function of the database. The spatial data is then stored, manipulated, and accessed in the same way as the standard, structured data **types** .

Universal Server has two other options that, although they don't fit into this discussion of extended **data types** , are also worth a mention. The Oracle OLAP Option (the Oracle Express Server) is for multidimensional OLAP analysis, and the Oracle Messaging Option is for email, documents, calendaring, scheduling, and directory services.

In Oracle, each table can be stored in a default form in a table space, as a cluster, or as a **hashed** cluster. In a cluster, rented tables are stored together in a pre-joined form. When the cluster has a **hashed** structure, it performs exact-match queries well. Oracle indexes, by default, have B-tree structures when stored in the standard table spaces or when used to index clusters. When an index is used to index a **hash** cluster, it obviously has a **hash** structure. Oracle7 Release 7.3 also has bitmapped indexes, which are better than conventional indexes for columns with a relatively small number of possible values, and are often used in OLAP-type applications.

Oracle lets you define multiple database triggers per table. The triggers can be defined to fire per operation or per affected row, before or after the operation. The triggers are coded in PL/SQL, a powerful 4GL-like language that includes a rich set of functions, user-defined **data types** (such as temporary tables), and cursors to process single or multiple rows at a tune. The triggers for the same event fire in any order, typically in creation order. If you need to enforce a firing order, the code can be combined in one trigger. An Oracle database trigger can have a WHEN clause, which is used to eliminate unnecessary trigger firing. Database triggers are not stored in a compiled form in Oracle 7.2. However, in Oracle 7.3, database triggers are stored in a compiled form, similar to that of stored procedures. This makes the implementation of Oracle triggers much more efficient. An Oracle trigger cannot query or perform a manipulation operation on a "mutating" table, where the trigger may see an inconsistent view of the table, because there is a transaction pending on the table (the firing transaction, in this case).

An Oracle stored procedure or function is a named set of PL/SQL statements that is stored in the database as a database object. You can execute a procedure interactively using an Oracle tool, such as SQL*Plus,

or call it explicitly in the code of a database application, such as an Orate Forms or embedded SQL application, or in the code of another procedure or trigger. When you create a stored procedure, Oracle compiles the procedure, stores the compiled code in memory, and stores the stored procedure in the database. When you invoke a stored procedure, Oracle verifies the procedure's validity and then executes the procedure. If the procedure is valid and currently in memory, the PL_/SQL engine simply executes the procedure's code. If the procedure is valid and not currently in memory, the PL_/SQL engine loads the compiled procedure from disk into memory and then executes it.

Queries

Oracle uses shared and exclusive locks to control concurrency. Each transaction acquires an exclusive data lock for each row it changes through an INSERT, UPDATE, or DELETE statement or a SELECT statement with the FOR UPDATE clause. It acquires a shared lock for each row read by a SELECT statement. Oracle also takes a table-level lock to prevent conflicting database definition changes from interfering with the users' transactions. Oracle never escalates locks; Oracle automatically performs locking to ensure data concurrency, data integrity, and statement-level read consistency. However, you can override the Oracle locking mechanisms per individual table or on an instance-wide level-- for example, for transaction-level read consistency ("repeatable reads") or transactions requiring exclusive access to specific resources.

The SQL language used in Oracle is compliant with the ANSI SQL-92 Entry Level standard. However, like all of the **other** DBMSs, it includes many extensions to the language to provide additional commands and additional functionality.

Oracle supports cursors using the ANSI SQL-92 standard syntax. In addition to fetching one row at a time, you can fetch an entire array of data at a time. However, you can move forward only through the cursor.

Oracle supports left and right outer joins using the wall-known Oracle outer join syntax. Left and right outer joins are specified by (+) symbols in the join condition of the WHERE clause of the SELECT statement, not in the FROM clause, as specified in the ANSI SQL-92 syntax.

Database Administration

Oracle7 Release 7.2 has a whole set of database administration tools that perform the following tasks:

* Database Manager starts and shuts down the database, customizes initialization parameters, and creates database aliases.
* User Manager creates user accounts and roles, grants privileges to users, and manages user passwords.
* Object Manager manages database objects such as tables, indexes, synonyms, and views, and it grants object privileges to users.
* Session Manager views and disconnects user sessions on the database.
* Database Expander expands the database to make room for additional data.
* Backup and Recovery Managers back up and recover your data.
* Password Manager changes the database system password from the database.

In Release 7.3, these features have been extended with a comprehensive toolset that provides a framework for managing the entire Oracle environment, with open interlaces and full integration into the leading SNMP-based network management frameworks. The Oracle Enterprise Manager console, which facilitates GUI-based systems management, is the central point of control for the Oracle environment. It consists of the following:

* Navigator is an object browser used to manage a tree structure of all of the nodes, databases, listeners, users, roles, and profiles.
* Map Window is a GUI view of the key objects; it manages and monitors subsets of the objects in the system. The subsets can be grouped

by any criteria.

* Job Scheduling System schedules, runs, and automates repetitive database tasks, even on remote sites.

* Event Management System is a monitor for database and system events, with a filtering mechanism to set thresholds and an interface with the Job Scheduling System to activate corrective tasks.

The Oracle Enterprise Manager Performance Pack is an additional set of products that DBAs can use for monitoring, diagnosing, and tuning the performance of large and diverse environments, databases, applications, and other configured events.

For highly secure systems, Oracle7 has the necessary features to enforce the required security policies. Oracle7 enforces data confidentiality through the following mechanisms:

* User Identification and Authentication establishes and verifies a user's identity before permitting a connection to the database.

* Discretionary Access Control (DAC) restricts users' ability to perform specific operations and access specified objects, based on privileges assigned to them. Authorized users can re-grant their rights to other users according to their discretion. Privileges can be granted per object, to a role, or across a system. A role is a user-defined group of privileges that can be granted to users or to other roles--it is a powerful mechanism for managing large numbers of users and privileges. Users can set 70 different types of security privileges.

* Mandatory Access Control (MAC) restricts users' access to data based on its sensitivity and the the user's clearance. To enforce MAC, the system must store and maintain security labels for each object and a clearance for each user.

* Accountability includes all of the actions on the system that are recorded and that can be tied to specific users. These actions are recorded in an audit trail that can be analyzed to detect security threats. Oracle7 provides auditing options per user, per database operation, per object accessed, and per system privilege.

* Database Encryption stores the database in an encoded form that can only be deciphered with the correct software and deciphering key.

For backup and recovery, the Oracle7 Enterprise Backup Utility works in conjunction with a media management product to help a DBA in two aspects of the backup and recovery process. The media management products offer reliable, high-performance handling of the backup media devices, and the backup utility ties in closely with the Oracle7 Server. The media management product can utilize multiple backup devices simultaneously by creating multiple data streams while performing compression and buffered I/O to improve the resource utilization. The utility can perform online and partial backup and restore functions. internet Support

The Oracle Universal Server includes the Oracle WebServer, which lets clients using Web browsers invoke stored procedures to generate dynamic Web documents. The WebServer lets you develop interactive, online applications that can change their behavior as the data in the underlying databases changes and then directly manipulate the data in those underlying databases.

Among the Leaders

While the company is obviously still carrying on with its relentless marketing drive, it seems that Oracle is finally putting its money where its mouth is with the latest Oracle7 release. Oracle7 Release 7.3 (basically Oracle8 without objects) is in line with the other relational DBMS products. There was a time when Oracle 6 technologically lagged behind some of its competitors, despite all its media hype. But with its Web support, partial and parallel database management facilities, high degree of parallelism-awareness, and numerous performance improvements, the Oracle7 releases--and in particular the Oracle Universal Server--are right up there among the technology leaders.

Martin Rennhackkamp is the owner and principal consultant of The Data

Base Approach, a corporation specializing in relational and distributed databases, based in Cape Town, South Africa. You can email Martin at rnr@dba.co.za.

* Oracle Corp., 500 Oracle Pkwy., Redwood Shores, CA 94065; 800-672-2537, 415-506-.7000, or fax 415-406-7200; http//www.oracle.com.

COMPANY NAMES: Oracle Corp.--Products
DESCRIPTORS: Software Single Product Review; DBMS
SIC CODES: 7372 Prepackaged software
TICKER SYMBOLS: ORCL
TRADE NAMES: Oracle7 7.3 (DBMS)--Evaluation; Oracle Universal Server (DBMS)--Evaluation
FILE SEGMENT: CD File 275

8/9/2 (Item 2 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)

01713459 SUPPLIER NUMBER: 16261887 (THIS IS THE FULL TEXT)
Getting integrity in SQL Server. (Sybase relational DBMS) (Enterprise Client/Server)
Mullins, Craig S.
DBMS, v7, n12, p99(3)
Nov, 1994
ISSN: 1041-5173 LANGUAGE: ENGLISH RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 2575 LINE COUNT: 00210

ABSTRACT: Sybase's SQL Server 10 relational DBMS (RDBMS) provides several features that support user-defined data integrity in its relational mode. User-defined integrity, along with referential and entity integrity, is inherent to the relational database model. This form of data integrity ensures that RDBMS properly handles the user data values that are stored within a database application. The SQL Server 10 features that support user-defined integrity are described and exemplified. They include check constraints, rules, unique constraints, user-defined data types, defaults and triggers. Database application developers should utilize these capabilities to provide system-managed data integrity, reusability and consistent data structures.
TEXT:

User-defined integrity is a component of the relational model that has been neglected for too long. Fortunately, several DBMS vendors have begun to implement more robust, user-defined integrity into their products. Foremost among these vendors is Sybase Inc. with its SQL Server System 10 product offering.

Most data professionals are aware that the relational model provides basic integrity features to support both referential integrity and entity integrity. However the concept of user-defined integrity is also inherent to the relational model. When user-defmed integrity is supported, the RDBMS can automatically manage the particular values that are stored within the **database** .

User-defined **integrity** constraints go far beyond simple **data type checking** and referential value **checking** . Values can be excluded from a specific column or colunms based upon business requirements. In the absence of user-defined integrity support, an application program typically performs this type of functionality. A systematic and non-bypassable method of integrity **checking** without the need to write code provides an obvious benefit in terms of **reduced** development time.

Types of User-Defined Integrity

Sybase SQL Server 10 provides user-defined integrity in several different ways, each of which I discuss in this article. The following features support user-defined integrity: **check** constraints, rules, unique constraints, user-defined **data types** , defaults, and triggers.

Check Constraints

A **check** constraint is a mechanism for allowing predicates to be defined on a column. The predicate is attached to the column as data definition language (DDL) and performs automatic edit **checking** of supplied values. Each **check** constraint is performed whenever data is inserted or updated. You can code check constraints at the column or table level.

Let's examine column-level constrains first. Column-level constraints consist of a name and the actual predicate:

```
create table employee
    (emp-id    int              not null,
    ssno       char(9)          not null,
```

```
   emp-name    varchar(50)        not null,
   salary      numeric(12.2)      not null
constraint salary-cons
check (salary < 50000.00).
   comm        numeric(12,2) null,
   bonus       numeric(9.2)  null)
```

Every constraint must have a name. Failure to specify a name explicitly causes SQL Server to generate a unique name for the constraint, which can be difficult to administer later. The name of the constraint in the above example is salary_cons. The predicate portion defines the actual conditions of the edit check and is coded as a typical SQL WHERE clause (without the actual "where" keyword, of course).

Unfortunately, you cannot define check constraints as a SELECT from another table. This limits their overall benefit. However, check constraints, even in this limited form, are superior to coding the condition into every application program that updates the column or columns in question.

In addition to colunuilevel check constraints, it is possible to specify check constraints at the table level. Instead of being attached to a single column, the constraint is attached to the entire table.

It is usually sufficient to code a check constraint at the column level. However, there are situations in which table-level check constraints are required. Any time two columns from the sane table must be specified in the constraint, a table-level check constraint is required.

The example below depicts a table-level check conaffit to ensure that an employee's bonus is less than or equal to his or her commission. It would have been impossible to code this particular constraint at the column level because it accesses two colunms (instead of one column and one constant as shown in the previous example).

```
create table employee
    (emp_id    int              not null,
    ssno       char(9)          not null.
    emp-name   varchar(50)      not null.
    salary     numeric(12.2)    not null,
    comm       numeric(12,2)    null,
    bonus      numeric(9.2)     null
 constraint do_not_payem
 check (bonus <= comm) )
```

You can attach user-defined messages to both columm- and table-level constraints. Consider the check constraint for the salary column the example. You can assign the message "salary too high" to the constraint by adding a message and binding it to the constraint as follows:

```
sp-bindmsg do_not_payem, 20005
```

Messages are always assigned a number greater than 20,000 because the first 19,999 message numbers are reserved for SQL Server use. The message text can be up to 255 characters long.

Finally, you can retrieve information on check constraints from the system using the sphelpconstraint system procedure. If you pass a table name as a parameter to sp_helpconstraint, a list of all constraints defined for the given table is displayed. This procedure is new in System 10.

Rules

Rules are similar to check constraints, but rules are "free-standing" database objects. You can create them using the "create rule" DDL statement, and they exist independently of any table or column. As with check constraints, you can use rules to define data validation. Once created, a rule can be bound to a table column. Thereafter, whenever data is inserted or updated, the system checks the rule to ensure that the data modification complies with it

The advantage of creating "free-standing" rules instead of using check constraints is enhanced reusability. Rules are reusable, check constraints are not. You can create and apply a free-standing rule as

foflows:

create rule state-rule as

@state in ("IL", " WI", "IN". "IA") exec sp_bindrule "state-rule", "authors.state"

The first statement creates the rule; the second binds it to a specific column in a specific table. You can bind the same rule to as many different columns in as many different tables as you desire. Moreover, keep in niind the following rules of thumb before binding a rule to a column:

* A column can have only one rule assigned to it SQL Server will, however, allow a rule to be bound to a column that already has a rule defined. In this case, the last rule bound to the column takes precedence.

* A column can have both a rule and a column-level check conswffit assigned to it. If the rule and the check constraint conflict, then you may have trouble.

* Rules are apphed whenever you insert or update data values.

* A rule that is no longer required can be removed from a column via the sp_unbindrule system procedure.

Check Constraints or Rules?

The second buffet in the previous list introduces an interesting problem. When should check constraints be used? When should rules be used? And what if both are used on the same column?

Sybase added check constraints to SQL Server System 10 to support the ANSI SQL standarxl. Ihere is no concept of a rule currently in the ANSI standard. SQL Server has featured implement for many releases. Both integrity features implement the same basic function: They place restrictions on the data values that can be stored in a column.

Of the two methods, rules are more flexible. Rules are created as free-standing database objects and can be bound to columns and user-defined data types. Check constraints, on the other hand, are specified in the table DDL. They are useful when a constraint exists between two columns of the same table.

In general, follow these guidelines:

* Use rules over check constraints when either would suffice and portability is not an issue. Rules are reusable and more flexible. * Use check constraints if you require conformance to the ANSI SQL standard or if you will be porting applications to and from environments and DBMSs. Many DBMS products support check constraints, but not rules. * Use check constraints when you want to compare two colunms of the same table. For example, if an employee's bonus must always be less than a percentage of his or her salary, tlle following check constraint would be approptiate:

check (bonus < salary * .10)

You cannot do this with a rule, because rules must always reference one variable (column) and one constant. * It is possible to define both a rule and a check constraint for a single column. If this occurs, be sure that the two are compatible. For example, avoid the following scenario:

check (state in 'IL', |PAI', |FLI') create rule state_rule as @state in (|GA', |CA', |LI') sp_bindrule state-rule, "table_name.state"

In this case, only rows specifying Illinois as the state could ever be inserted into the table.

Unique Constraints

SQL Server System 10 also supports unique constraints. You can apply this type of constraint to a column or columns to ensure that duplicate values cannot be stored.

SQL Server enforces uniqueness by automatically creating a unique index on the specified column or columns. Therefore, unique constraints can also specify the type of index to be generated: clustered or nonclustered. Nonclustered is the default. A unique constraint will allow the column(s) to store one null.

Although the relational model forbids duplicate rows, most relational DBMS products let you store duplicates by default. Unique constraints enable the database designer to force the DBMS to follow this relational

tenet.

User-Defined Data Types

All RDBMS products provide basic system data types such as integer, character, and decimal. SQL Server, however, lets users define additional data types. User-defined data mxs are based on system-defined data types, but can provide additional constraints on the data content. For example, a user-defined data type can provide a precision, scale, or length attribute, as well as a column property (that is, null or not null). User-defined data types, once created in the database, become fundamental data types, which any table in that database can use - just like a system-defined data type. Rules and defaults can be bound to user-defined data types.

An example of a user-defined data type definition follows:

sp_addtype proper_name, "char(40)", "null"

User-defined data types are quite useful for ensuring consistency throughout a database design. Consider, for example, a system in which it is necessary to store a social security number in multiple tables. Confusion may arise as to whether the number should be stored in character or numeric format Furthermore, if you store it in character format should it contain embedded hyphens? See Table 1 for a list of valid options for storing social security numbers.

On the other hand, you can define a user-defined data type, such as SSN. The SSN data will be standard and can be used for all columns that store social security number data. This approach ensures consistency from table to table and colunm to column. Whether the user-defined data type is character or numeric is not important The point is that the data definition is consistent.

An additional benefit of establishing user-defined data types is a higher level of abstraction in a database design. It is much easier to discuss the salary data type (with all its implied definitions, properties, and constraints) than it is to talk about a decimal(12,2) or smallmoney data type (with no implied characteristics other than its inherent type).

User-defined data types can also decrease maintenance. When a default or rule is bound to a user-defined data type, every column that is assigned that user-defined data type "inherits" the attached default and/or rule. Likewise, when a developer changes a rule or default bound to a user-defined data type, the change is automatically reflected in all columns that are assigned that user-defined data type.

TABLE I

Data Type	Example
char(11)	"123-45-6789"
char(9)	"123456789"
integer	123456789
decimal	123,456,789

Social security number swrage options.

Supporting Domains Using SQL Server

Domains have been a part of the relational model since its first definition. However, no current RDBMS explicitly supports domains. SQL Server 10 supports domains only implicitly and incompletely.

What is a domain? According to Chris Date: A domain is a set of possible data values of some particular by from which fields in the database draw their actual values" (from Relational Database Selected Writings, Addison-Wesley, 1986). SQL Server's domain support is only partial because it does not support the following domain characteristics:

* user-defined comparison operators; * limiting comparison operators by domain; and * checking to ensure that two columns to be compared are pooled from the same (or compatible) domains.

You can partially implement domains in SQL Server using a combination of user-defined data types, rules, and defaults. Suppose you want to define

a domain for product codes to be stored in a SQL Server database. All product codes conform to the following standards:

* product codes are six bytes long; * a product code must begin with an alphabetic character; * the second byte must be numeric (but cannot be 0), the next three bytes can be anything, and the last byte must be either "@" or "#"; and * if a product code is unknown, it should default to null (unless it is the primary key or a part of the primary key).

To implement a domain for the product code, take the foflowing steps:

1. Create a user-defined data type, such as prodcode:
sp_addtype prodcode, "char(6)", "null"
2. Create a rule, such as prodcode-rule:
create rule prodcode_rule as @prodcode like "[A-Z] [1-91] - [#.@]"
3. Create a default, such as prodcode-deflt:
create default prodcode_deflt as NULL
4. Create all columns containing product code infonnation, specifying the "prodcode" user-defined data type. If the column participates in a primary key, specify the "not null" property directly in the table to override the property in directly user-defined data type. Bind the prodcode_rule and the prodcode_deflt to all colunms containing product codes.

Remember, however, that this provides rudimentary domain support only. It does not implement full domain support as described in Dr. E. F. Codd and C. J. Date's relational writings.

Triggers

It is also possible to support user-defined integrity in SQL Server through the use of triggers. Triggers are event-driven, specialized procedures stored in the RDBMS. Each trigger is attached to a single, specified table. You can think of a trigger as an advanced forrn of rule" or constraint" written using procedural logic. You cannot call or execute a trigger directly; rather, it is automatically executed (or fired") by the RDBMS as the result of an action - usually a data modification to the associated table.

Although triggers are often used for implementing referential integrity, there are many practical reasons for using triggers to iinplement user-defined integrity. Quite often it is impossible to code business rules into the database using only DDL For exarnple, the business rule may be too complex to support using a rule or check constraint. Triggers offer a flexible vehicle for the specification of user-defined integrity. Complex strings of instructions can be coded and stored within the DBMS as a trigger. Whenever data is added, removed, or modified, the system wfll execute the logic in the trigger to ensure that required integrity constraints are maintained.

On to Integrity

SQL Server System 10 provides a wealth of mechanisms for supporting user-defined relational integrity. Developers would be wise to implement these features to develop robust applications that provide system-managed data integrity, promote reusability, and provide consistent data structures.

* Sybase Inc., 6475 Chfistie Ave., Emeryville, CA 94608; 510-922-3500 or fax 510-922-4436.

SPECIAL FEATURES: illustration; table
COMPANY NAMES: Sybase Inc.--Products
DESCRIPTORS: Database Application Development Software; Utilization; Tutorial; Programming Instruction; Data Integrity; Methods
SIC CODES: 7372 Prepackaged software
TRADE NAMES: Sybase SQL Server 10 (Database application development software)--Usage
FILE SEGMENT: CD File 275

```
Set     Items   Description
S1    1796940   VERIFY??? OR VERIFI?????? OR CHECK??? OR VALIDAT???
S2    9159632   COMPAR???? OR MATCH??? OR EQUAL??? OR UNEQUAL??? OR SYNCHR-
            ON???? OR UNSYNCHRON???? OR ASYNCHRON????
S3      15954   (DATABASE OR REFERENTIAL OR ENTITY OR KEY OR DOMAIN OR COL-
            UMN OR TABLE)(3N)INTEGRITY OR DATABASE(3N)(CHANGE OR CHANGING
            OR CHANGED OR UPDAT??? OR ALTER??? OR MODIFY OR MODIFI?????? -
            OR EDIT??? OR VIOLAT??? OR CORRUPT??? OR COMPROMIS???)
S4    3565126   HASH??? OR MD5 OR CRC32 OR COMPRESS??? OR GZIP OR BZIP2 OR
            WINZIP OR ZIP OR REDUCED OR COMPACT OR MINIATURE OR ABRIDGED -
            OR SUMMARIZ???
S5   16719513   METADATA? ? OR META()DATA OR ATTRIBUTE? ? OR FIELD? ? OR P-
            ROPERT??? OR DATAFOUNDRY OR UML OR SQL92 OR DATA(2N)TYPE? ? OR
             SIZE OR LENGTH OR FILENAME? ? OR (FILE OR RECORD)()NAME? ? OR
             PARAMETER? ? OR METAFILE? ?
S6         51   S1 AND S2 AND S3 AND S4 AND S5
S7         33   S6 AND (PY<2003 OR PD<20020707)
? show files
File   2:INSPEC 1898-2006/Jan W1
         (c) 2006 Institution of Electrical Engineers
File   6:NTIS 1964-2006/Jan W3
         (c) 2006 NTIS, Intl Cpyrght All Rights Res
File   8:Ei Compendex(R) 1970-2006/Jan W3
         (c) 2006 Elsevier Eng.  Info. Inc.
File  34:SciSearch(R) Cited Ref Sci 1990-2006/Jan W4
         (c) 2006 Inst for Sci Info
File  35:Dissertation Abs Online 1861-2006/Jan
         (c) 2006 ProQuest Info&Learning
File  56:Computer and Information Systems Abstracts 1966-2006/Jan
         (c) 2006 CSA.
File  57:Electronics & Communications Abstracts 1966-2006/Jan
         (c) 2006 CSA.
File  60:ANTE: Abstracts in New Tech & Engineer 1966-2006/Jan
         (c) 2006 CSA.
File  65:Inside Conferences 1993-2006/Jan W4
         (c) 2006 BLDSC all rts. reserv.
File  94:JICST-EPlus 1985-2006/Nov W2
         (c)2006 Japan Science and Tech Corp(JST)
File  95:TEME-Technology & Management 1989-2006/Jan W4
         (c) 2006 FIZ TECHNIK
File  99:Wilson Appl. Sci & Tech Abs 1983-2005/Dec
         (c) 2006 The HW Wilson Co.
File 111:TGG Natl.Newspaper Index(SM) 1979-2006/Jan 24
         (c) 2006 The Gale Group
File 144:Pascal 1973-2006/Jan W1
         (c) 2006 INIST/CNRS
File 434:SciSearch(R) Cited Ref Sci 1974-1989/Dec
         (c) 1998 Inst for Sci Info
File 636:Gale Group Newsletter DB(TM) 1987-2006/Jan 27
         (c) 2006 The Gale Group
```

STIC Search Results Feedback Form

EIC 2100

Questions about the scope or the results of the search? Contact ***the EIC searcher or contact:***

Anne Hendrickson, EIC 2100 Team Leader
272-3490, RND 4B28

Voluntary Results Feedback Form

- *I am an examiner in Workgroup:* ☐ *Example: 2133*
- *Relevant prior art **found**, search results used as follows:*
  - ☐ 102 rejection
  - ☐ 103 rejection
  - ☐ Cited as being of interest.
  - ☐ Helped examiner better understand the invention.
  - ☐ Helped examiner better understand the state of the art in their technology.

Types of relevant prior art found:

- ☐ Foreign Patent(s)
- ☐ Non-Patent Literature
  (journal articles, conference proceedings, new product announcements etc.)

- *Relevant prior art **not found:***
  - ☐ Results verified the lack of relevant prior art (helped determine patentability).
  - ☐ Results were not useful in determining patentability or understanding the invention.

Comments:

Drop off or send completed forms to STIC/EIC2100 RND, 4B28